

# Part II: Algorithms and Applications

Speaker: Jeffrey Flanigan

# Intro

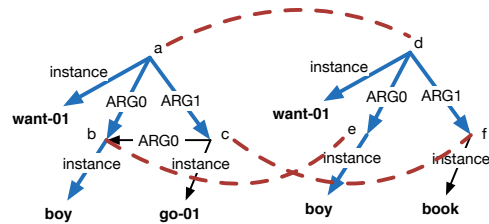
- You know how to annotate AMRs
- Now, we want to use them!
- To use AMRs, we need automatic **parsers**
- But first: **alignment**
- **Evaluation** (inter-annotator agreement, parser output)
- And also:
  - **Graph grammars** (like CFGs, but for graphs)
  - **Applications**

# Alignment

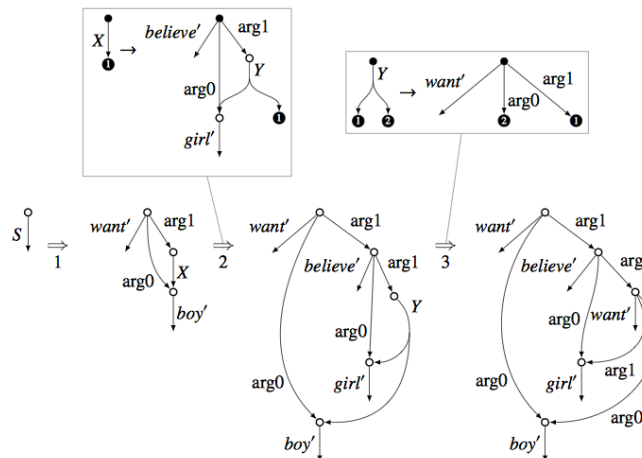
IAEA accepted North Korea's proposal in November.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# Evaluation

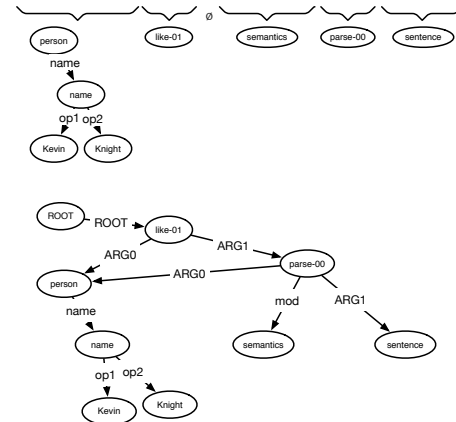


# Graph grammars

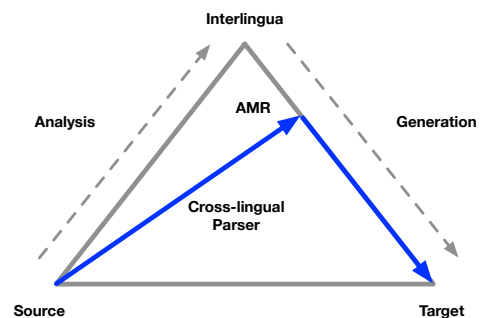


# Parsing

Kevin Knight likes to semantically parse sentences



# Applications



# Outline

- Alignment
- Parsing
- Evaluation
- Graph Grammars and Automata
- Applications

# Alignment: Motivation

- AMR annotation has no explicit alignment to sentence
- Training data has whole sentence – AMR graph pairs
- For generalization performance, need fine-grained correspondence between words and pieces of AMR
- Alignments provide this correspondence

Need alignments to train parsers, etc

# Alignment

The **tour** was a **surprise offer** made by **North Korea** in **November**.

```
(t / thing
:ARG0-of (s / surprise-01)
:ARG1-of (o / offer-01
          :ARG0 (c / country
                  :name (n / name
                        :op1 "North"
                        :op2 "Korea"))
          :time (d / date-entity
                 :month 11))
:domain (t2 / tour-01))
```

- Align concepts with words
- Can also align edges with function words

# Alignment

- Alignment
  - Motivation
  - **JAMR's rule-based aligner**
  - ISI EM aligner
- Parsing
- Evaluation
- Graph Grammars and Automata
- Applications

## JAMR Aligner (Flanigan et al, 2014)

- Aligns graph fragments to spans of words (edges not in fragments are unaligned)
- Uses a set of handcrafted rules
- Uses lemmatizer, string edit distance to match concepts with words
- Rules for: named entities, date entities, special concepts, negation, degrees, etc (15 total rules)



# JAMR Aligner

For each rule

- Greedily align concepts in a depth first traversal of the AMR graph
- Rules are applied in a specified order

# JAMR Aligner

**IAEA accepted North Korea 's proposal in November.**

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# JAMR Aligner

## Rule 1) Date entity

IAEA accepted North Korea 's proposal in **November**.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# JAMR Aligner

## Rule 3) Named entity

**IAEA** accepted **North Korea** 's proposal in **November**.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# JAMR Aligner

Rule 5) Single concept (use lemma)

IAEA accepted North Korea 's proposal in November.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# JAMR Aligner

Rule 6) Fuzzy single concept (longest string prefix > 4)

IAEA accepted North Korea 's proposal in November.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# JAMR Aligner

Rule 10) person-of/thing-of

IAEA accepted North Korea 's proposal in November.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# JAMR Aligner

IAEA accepted North Korea 's proposal in November.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```



# JAMR Aligner

Evaluate on 200 hand-aligned sentences:

$F_1$ : 90%

Precision: 92%

Recall: 89%

## Extracted concept table

8	critical => (critical)
2	critical => (criticize-01)
1	critically => (critical)
1	criticised => (criticize-01)
14	criticism => (criticize-01)
30	criticized => (criticize-01)
1	critics => (critic)
4	critics => (person :ARG0-of (criticize-01))
5	crop => (crop)
5	crops => (crop)
3	cross => (cross)
15	cross => (cross-02)
3	cross => (cross-border)
2	cross => (cross-strait)
1	crossed => (cross-00)
2	crossing => (cross-02)

# ISI Aligner (Pourdamghani et al, 2014)

- Aligns each concept or edge to at most one word
- Learns from data using EM
- Inspired by MT alignment models
- Basic idea: convert graph to linear string, use word alignment model

# ISI Aligner

**IAEA accepted North Korea 's proposal in November.**

```
(a / accept-01
  :ARG0 (o / organization
    :name (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# ISI Aligner

**IAEA accepted North Korea 's proposal in November.**

```
accept-01 :ARG0 organization :name name :op1  
"IAEA" :ARG1 thing :ARG1-of propose-01 :ARG0  
country :name name :op1 "North" :op2 "Korea" :time  
date-entity :month 11
```

Linearize the AMR using a  
depth-first traversal

# ISI Aligner

**IAEA accepted North Korea proposal in November**

```
accept organization name IAEA thing propose-01  
country name North Korea :time date-entity 11
```

English: remove stop words

AMR: remove special concepts, relations that don't  
usually align, quotes, and sense tags

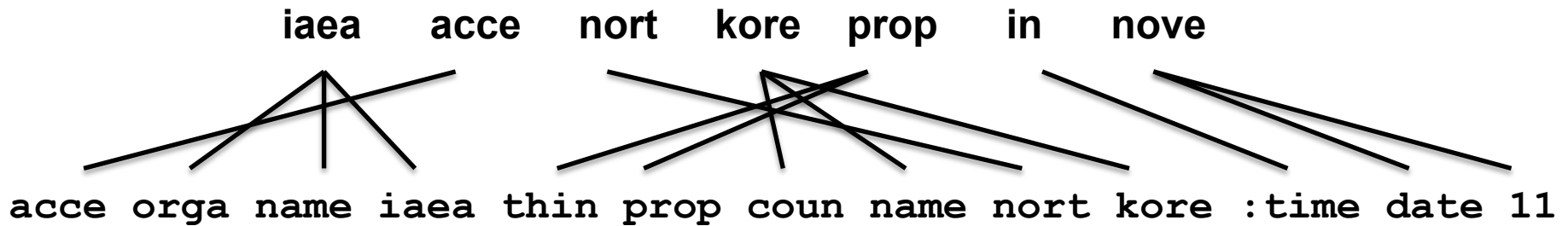
# ISI Aligner

iaea acce nort kore prop in nove

acce orga name iaea thin prop coun name nort kore :time date 11

Both: Lowercase and stem to the first four letters

# ISI Aligner



Run IBM alignment models with a symmetrization constraint, and project to AMR graph

**Alignments are 1-to-many**

# Alignment: Summary

	JAMR aligner	ISI aligner
Alignment type	Graph fragment to span of words	Each concept or edge to at most one word
Aligns edges	No	Yes
Learned from data	No	Yes
Gold standard available	<a href="https://github.com/jflanigan/jamr">https://github.com/jflanigan/jamr</a>	<a href="http://amr.isi.edu/research.html">http://amr.isi.edu/research.html</a>
$F_1$ score*	90%	83%

\*not directly comparable, since different gold standard

In general, the desired type of alignment will depend on the application

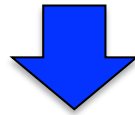


# Parsing

- Alignment
- **Parsing**
  - Graph-based parsing
    - Structured prediction
    - Concept identification
    - Relation identification
      - Maximum spanning connected graph algorithm (MSCG)
      - Graph determinism constraints using Lagrangian relaxation
    - Experiments
  - Transition-based parsing
  - Parsing using syntax-based MT
- Evaluation
- Graph Grammars and Automata
- Applications

# Parsing

Kevin Knight likes to semantically parse sentences.



```
(1 / like-01
  :ARG0 (p / person
    :name (n / name
      :op1 "Kevin"
      :op2 "Knight"))
  :ARG1 (p2 / parse-00
    :ARG0 p
    :ARG1 (s / sentence)
    :mod (s2 / semantics)))
```

# JAMR Overview (Flanigan et al, 2014)

Input

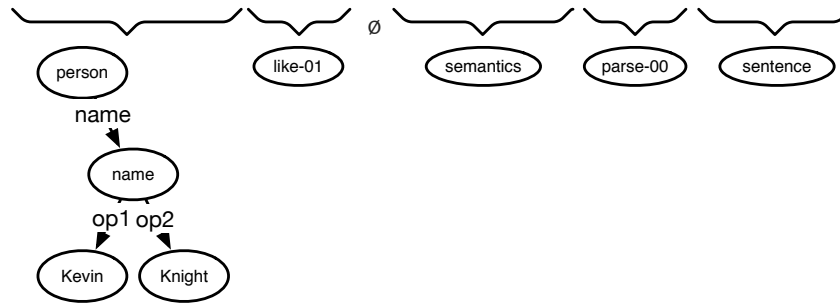
Kevin	Knight	likes	to	semantically	parse	sentences
-------	--------	-------	----	--------------	-------	-----------

# JAMR Overview

Input

Kevin	Knight	likes	to	semantically	parse	sentences
-------	--------	-------	----	--------------	-------	-----------

Concept ID

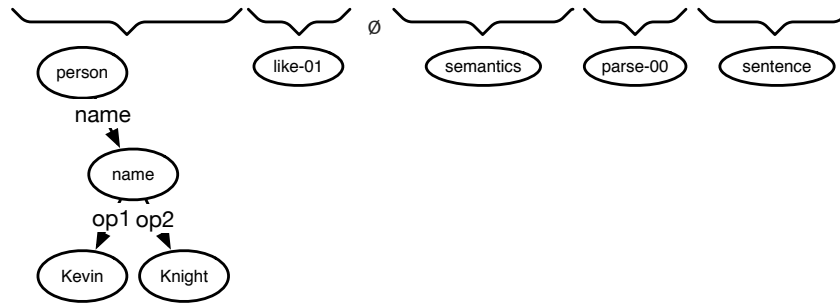


# JAMR Overview

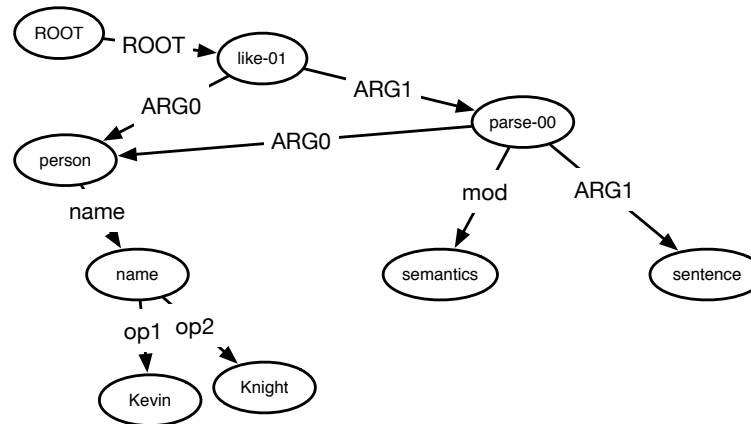
Input

Kevin	Knight	likes	to	semantically	parse	sentences
-------	--------	-------	----	--------------	-------	-----------

Concept ID



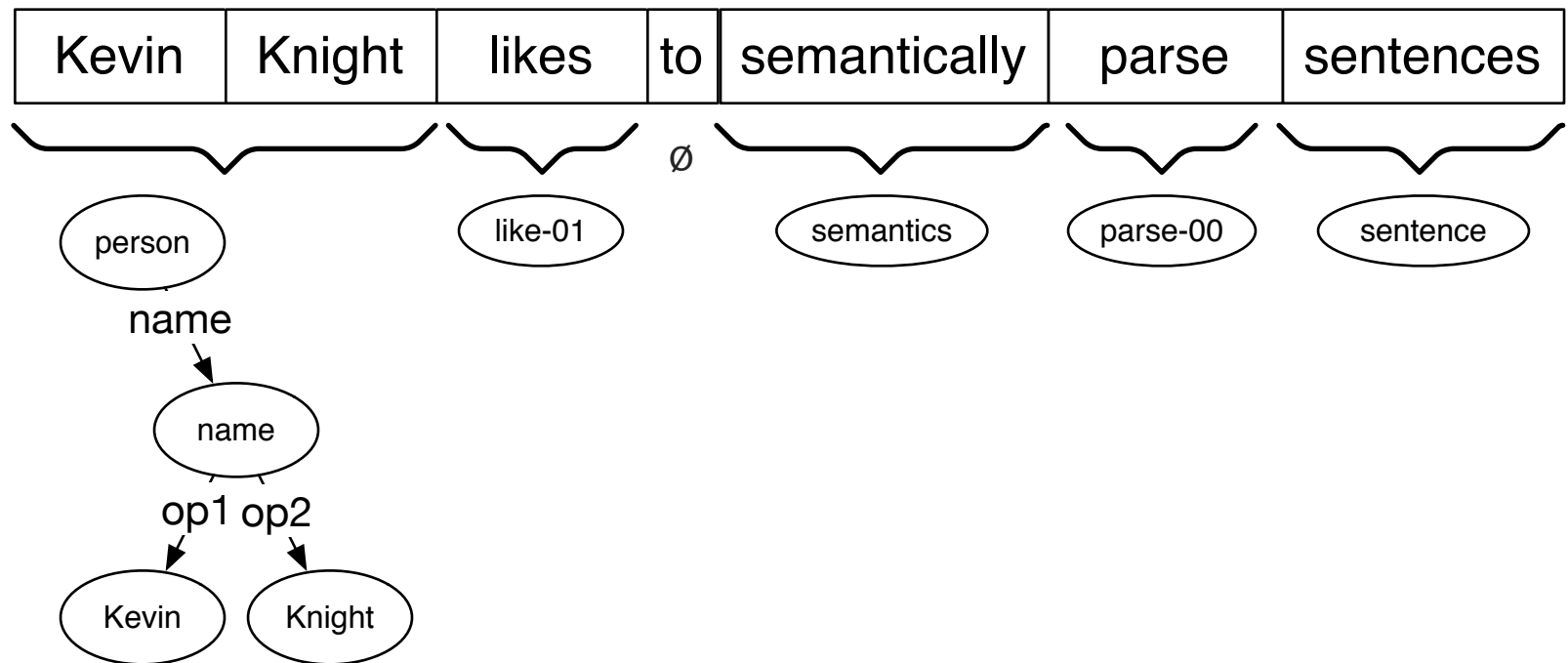
Relation ID



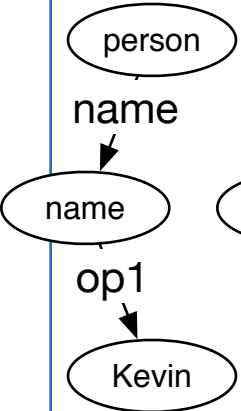
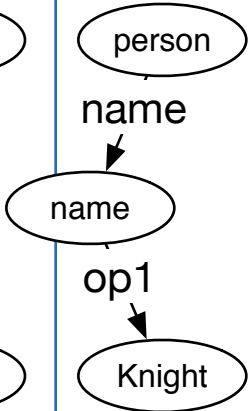
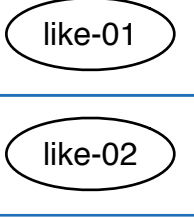

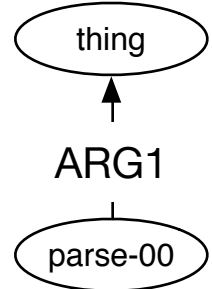

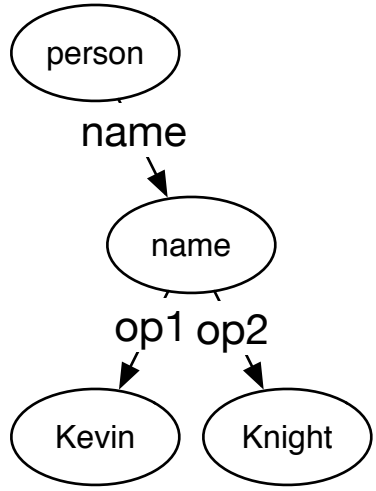
# Concept Identification

Kevin	Knight	likes	to	semantically	parse	sentences
-------	--------	-------	----	--------------	-------	-----------

# Concept Identification



# Concept Identification

Kevin	Knight	likes	to	semantically	parse	sentences
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
						
						

## Extracted concept table

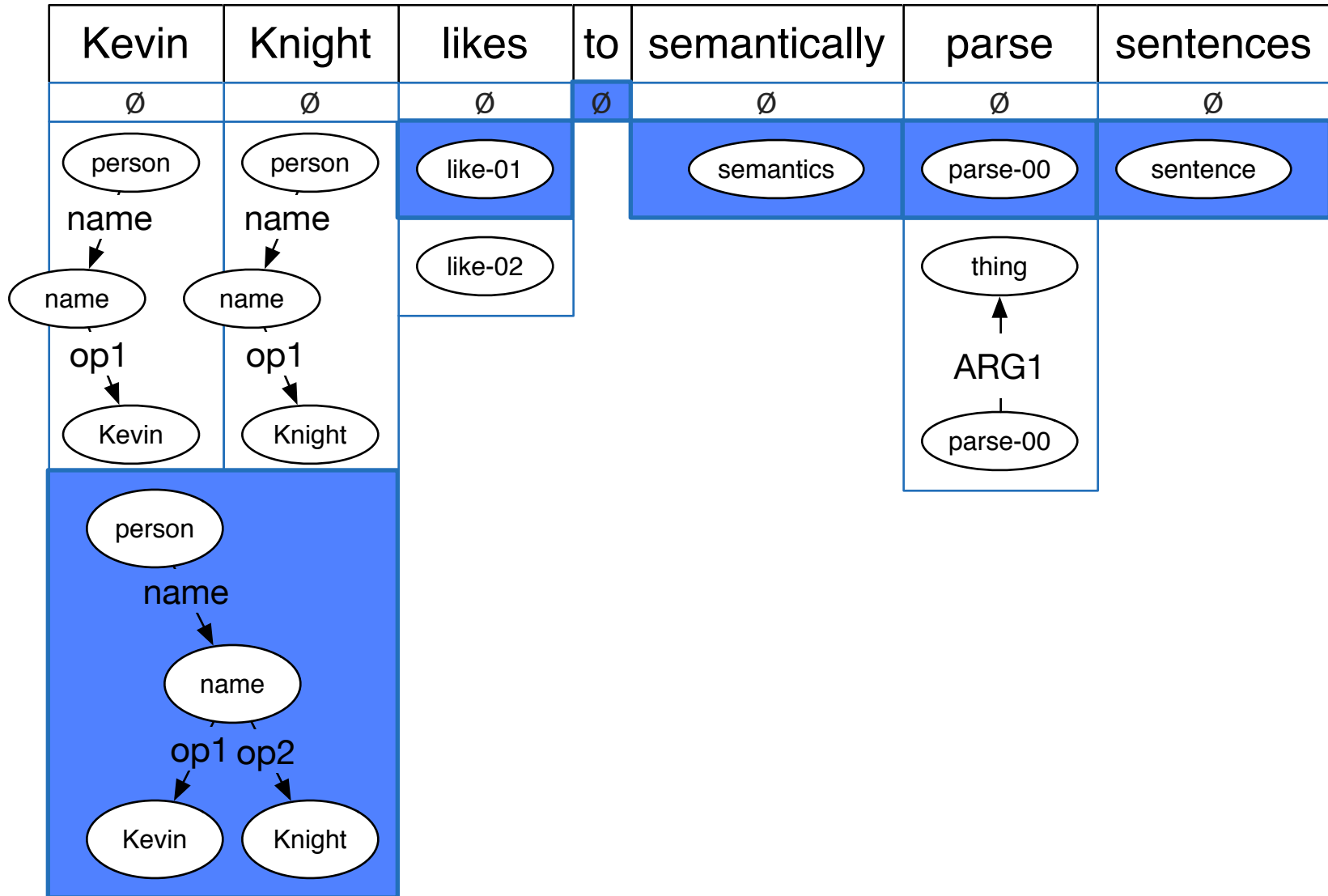
```

7      likes => like-01
1      likes => like-02
2      parse => parse-00
1      parse => (thing :ARG1-of (parse00))
1      semantically => semantics
2      sentences => sentence

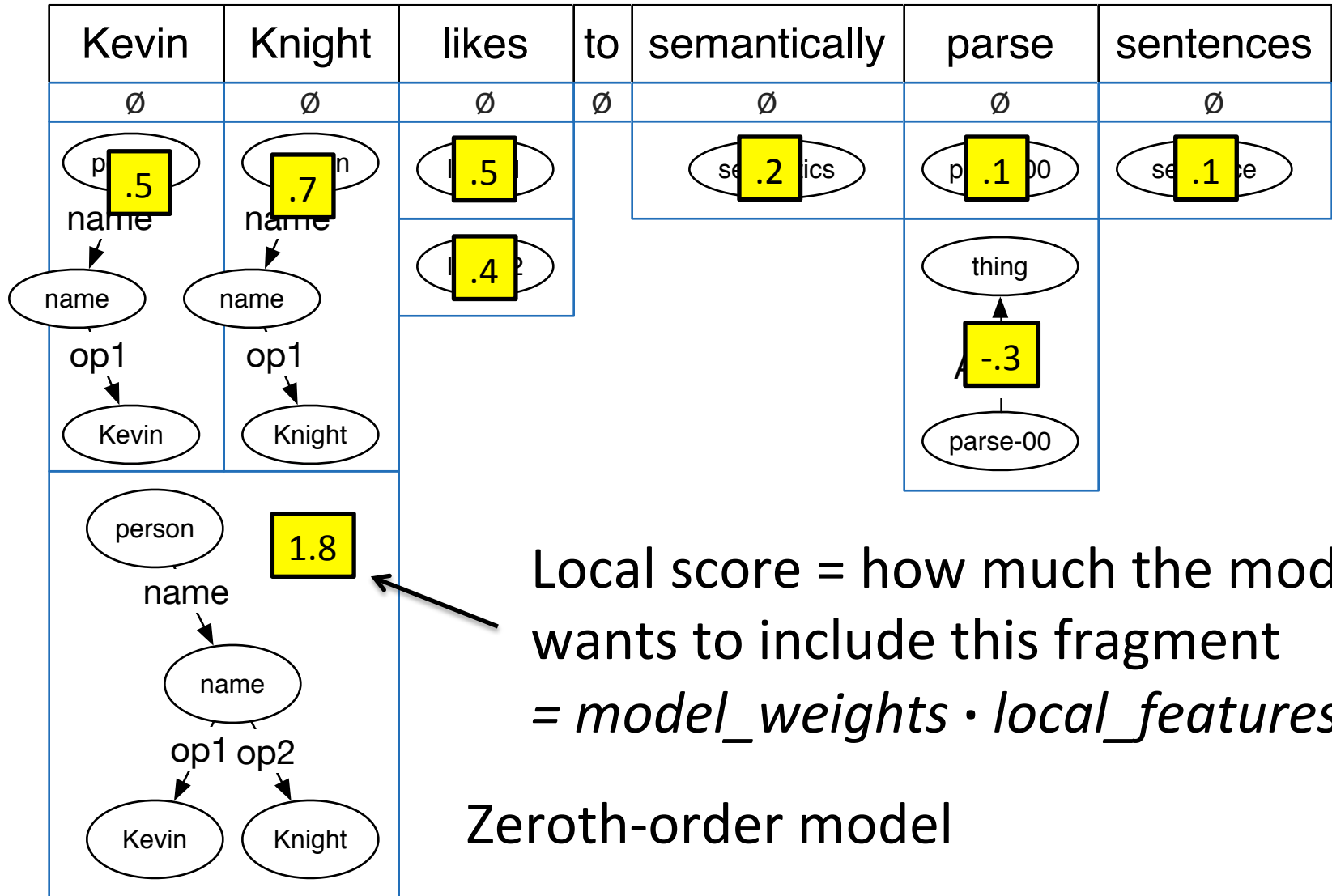
```



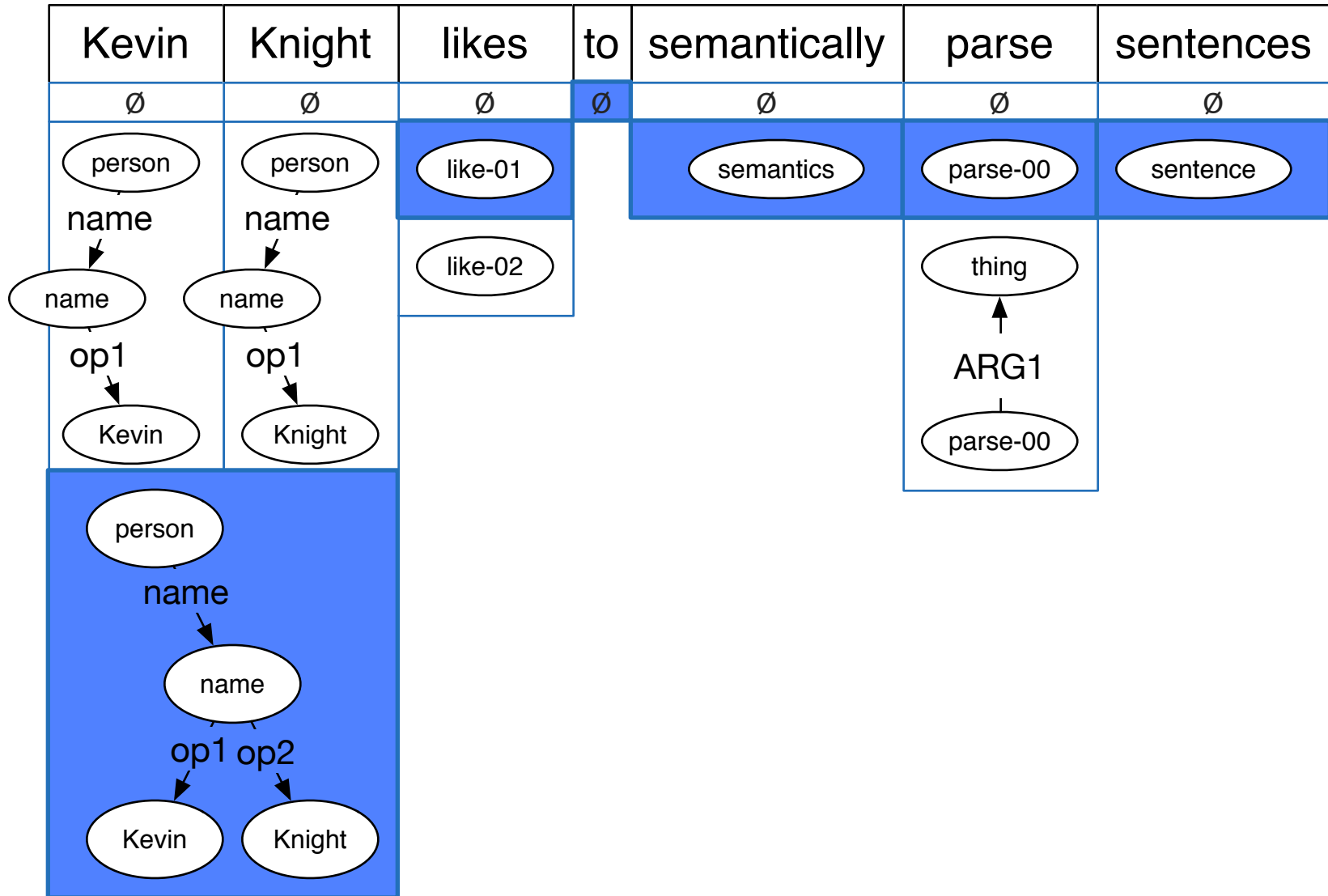
# Concept Identification



# Concept Identification



# Concept Identification



# Training

- AdaGrad structured perceptron

Learning rate

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{\sum_{t'=1}^t g_i^{t'}}} g_i^t$$

Model weight component i at step t+1

Gradient

# Training

- AdaGrad structured perceptron

Learning rate

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{\sum_{t'=1}^t g_i^{t'}}} g_i^t$$

Model weight component  $i$  at step  $t+1$

Gradient

$$g_i^t = feat_i(X_t, \hat{Y}_t) - feat_i(X_t, Y_t)$$

Input

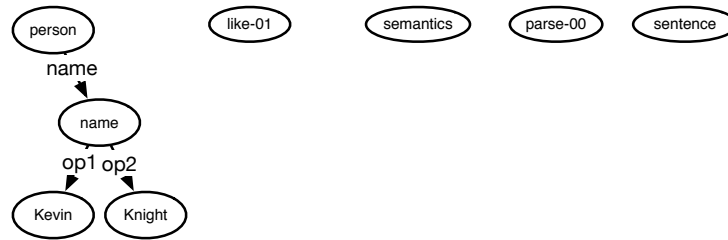
Predicted output

Gold output

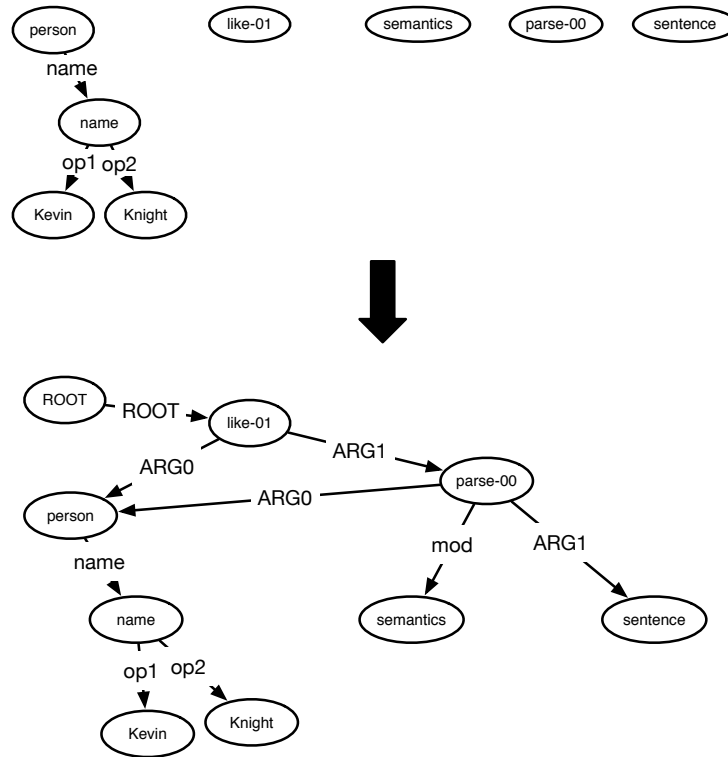
# Relation Identification

- Evaluation
- Alignment
- Parsing
  - Graph-based parsing
    - Concept identification
    - **Relation identification**
      - Maximum spanning connected graph algorithm (MSCG)
      - Graph determinism constraints using Lagrangian relaxation
    - Experiments
  - Transition-based parsing
  - Parsing using syntax-based MT
- Graph Grammars and Automata
- Applications

# Relation Identification

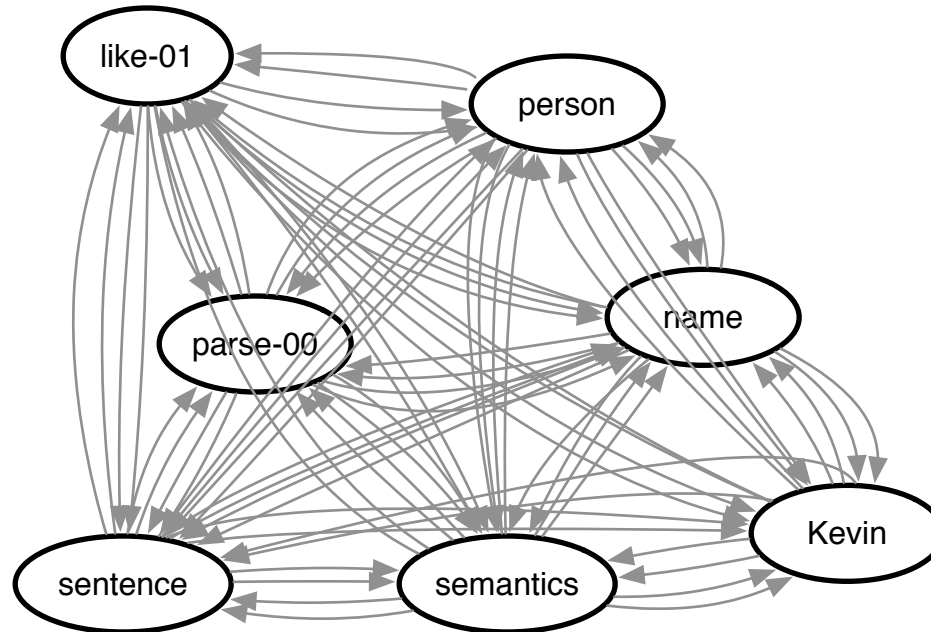


# Relation Identification

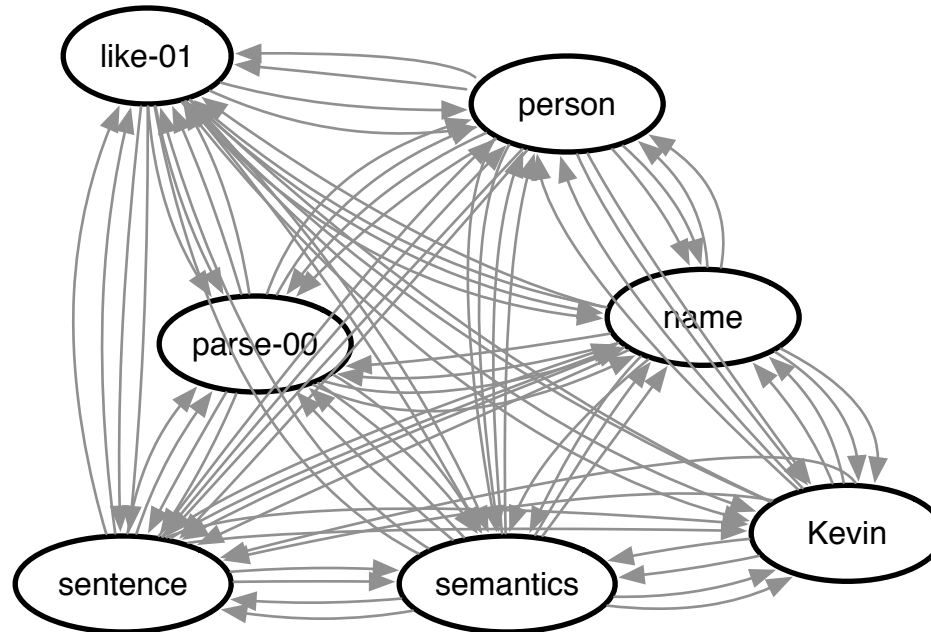




# Dense Graph

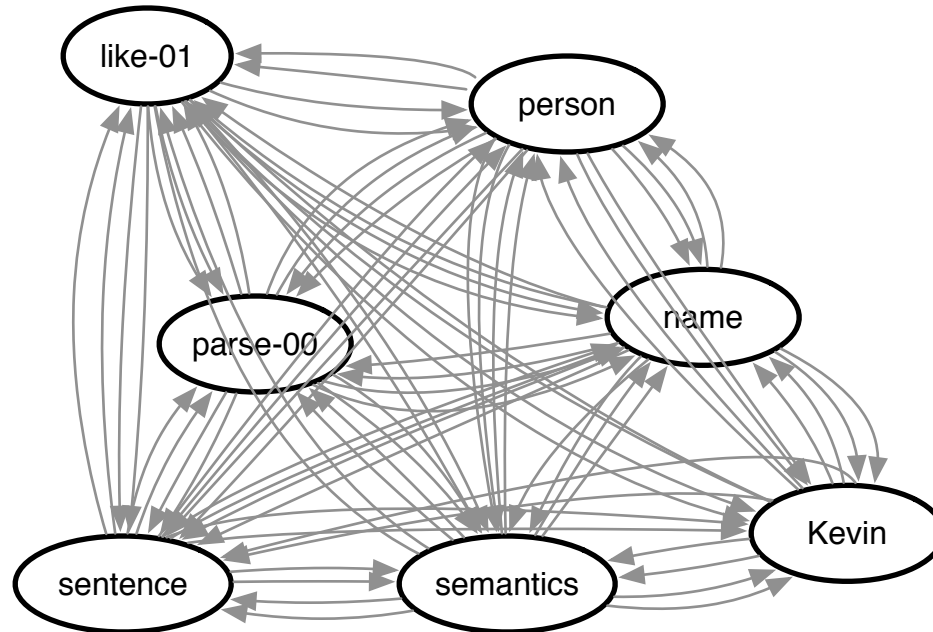


# Dense Graph



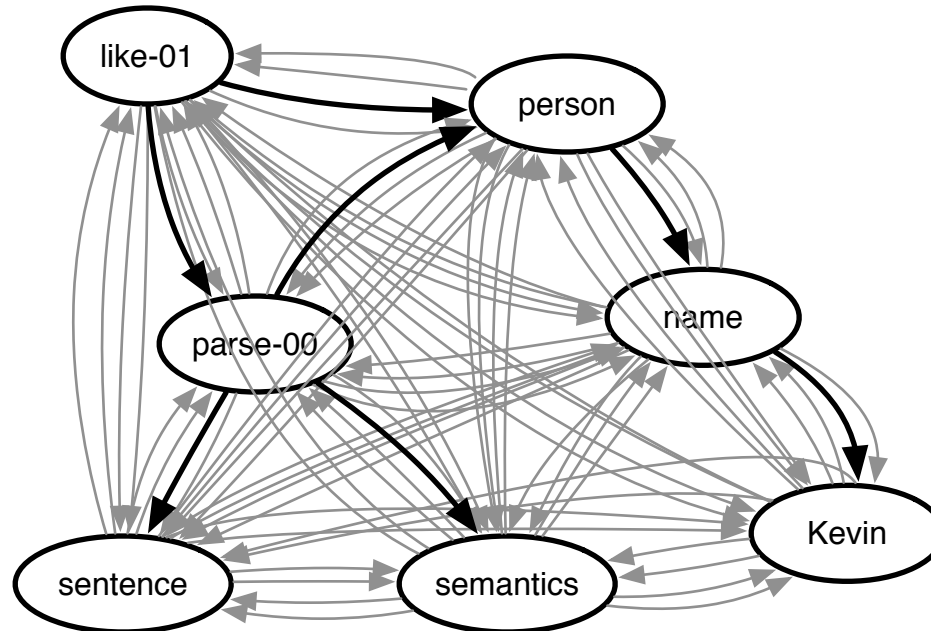
- All possible edges between all nodes
- Edges w/ weights

# Dense Graph



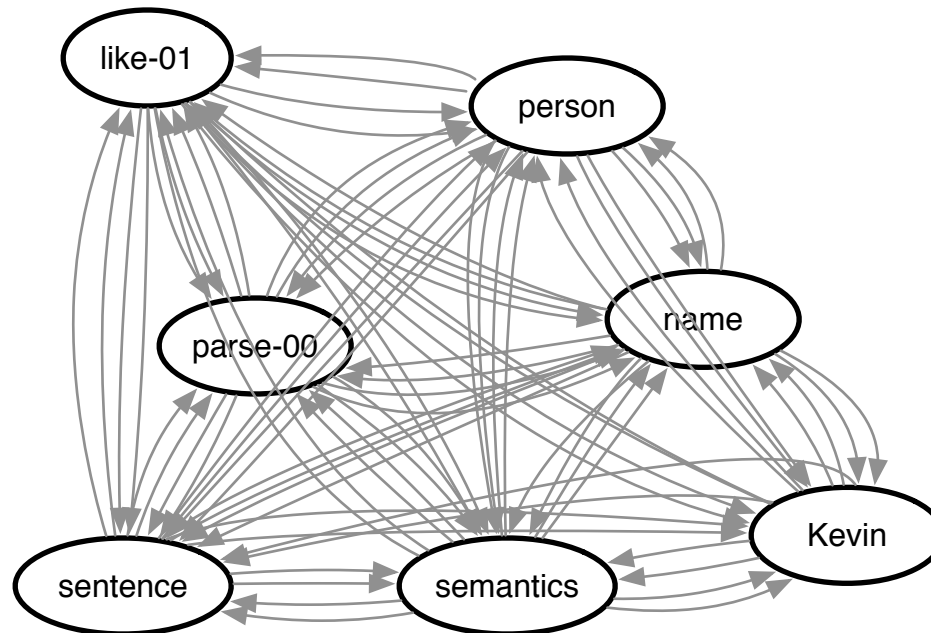
Edge weight = how much the model wants to include that edge in the output graph

# Dense Graph



Output graph = max subgraph with constraints on well-formedness

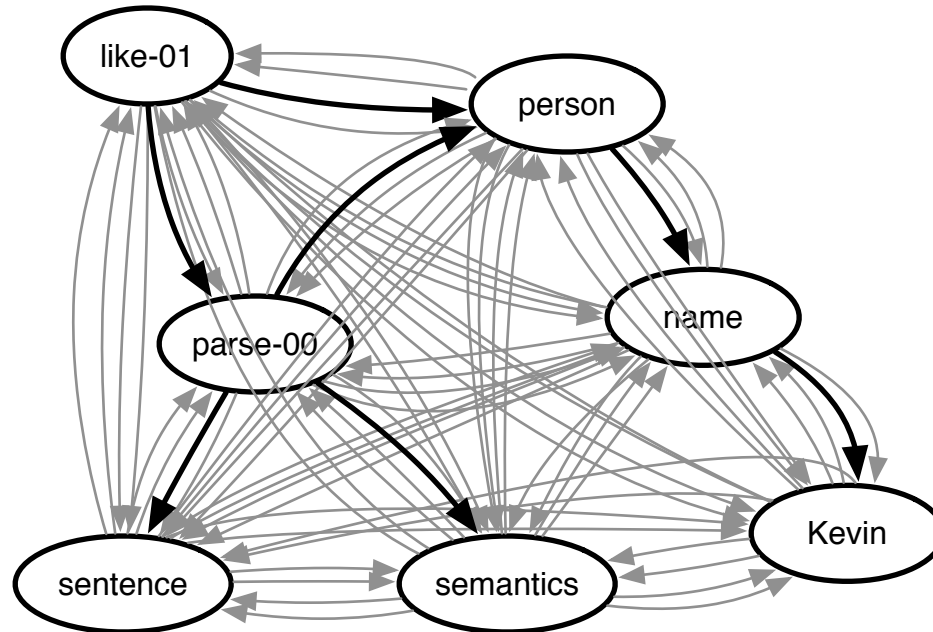
# Dense Graph



$\mathbf{Z}$  binary vector, indicates which edges are selected

$\phi$  real vector, contains the edge weights

# Max Subgraph



Relation ID  
optimization problem

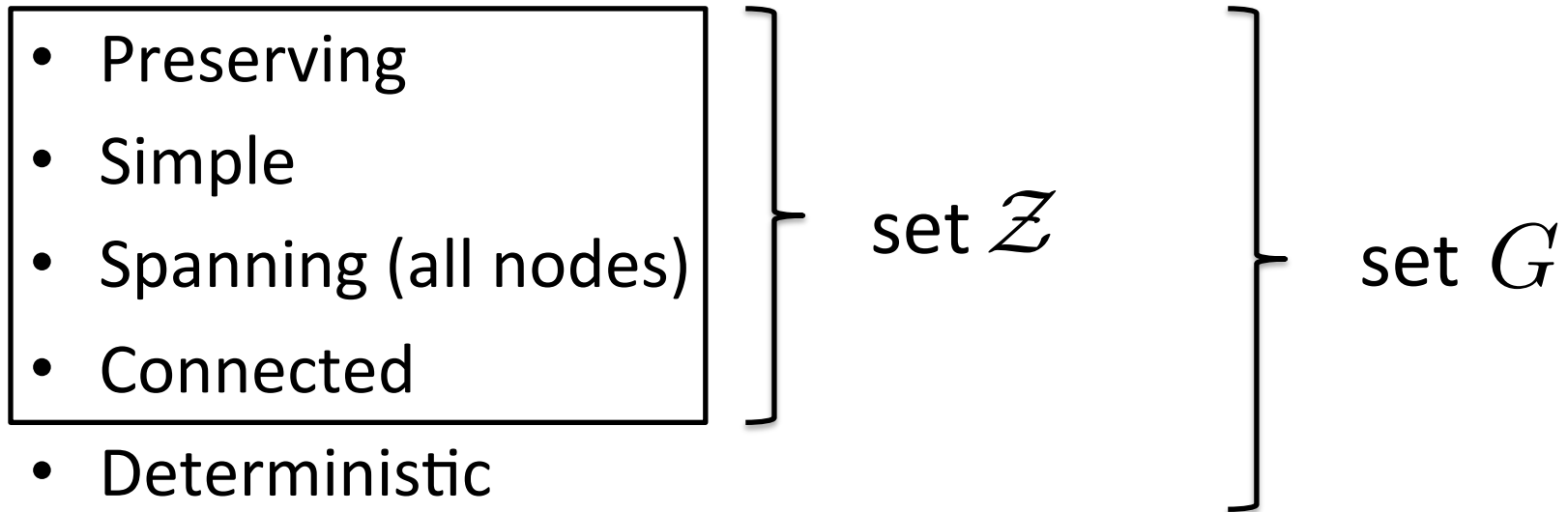
$$\max_{\mathbf{z} \in G} \phi^T \mathbf{z}$$

Set of graphs satisfying the constraints

# Output Graph Properties (Constraints)

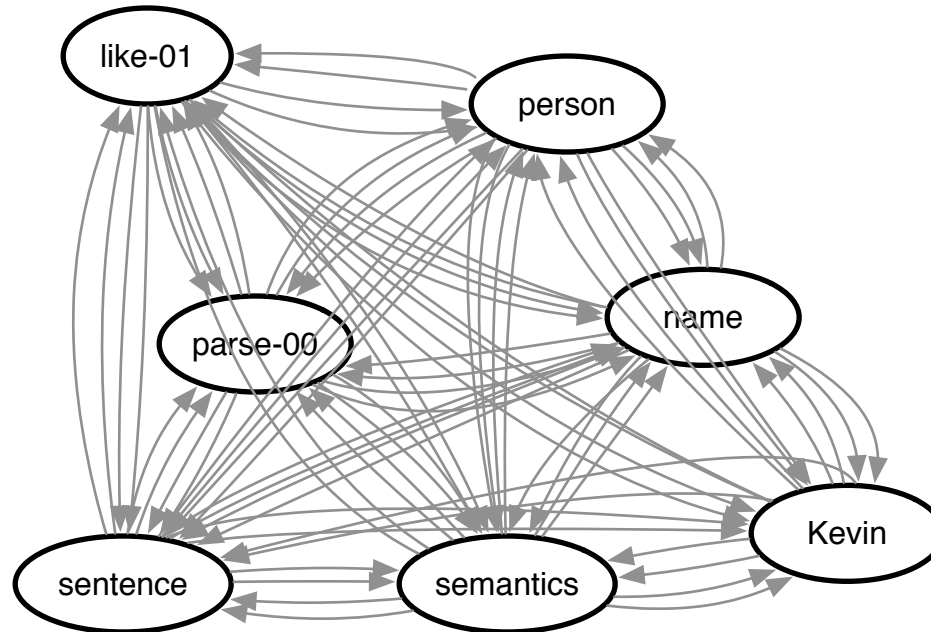
- Preserving
- Simple
- Spanning (all nodes)
- Connected
- Deterministic

# Output Graph Properties (Constraints)

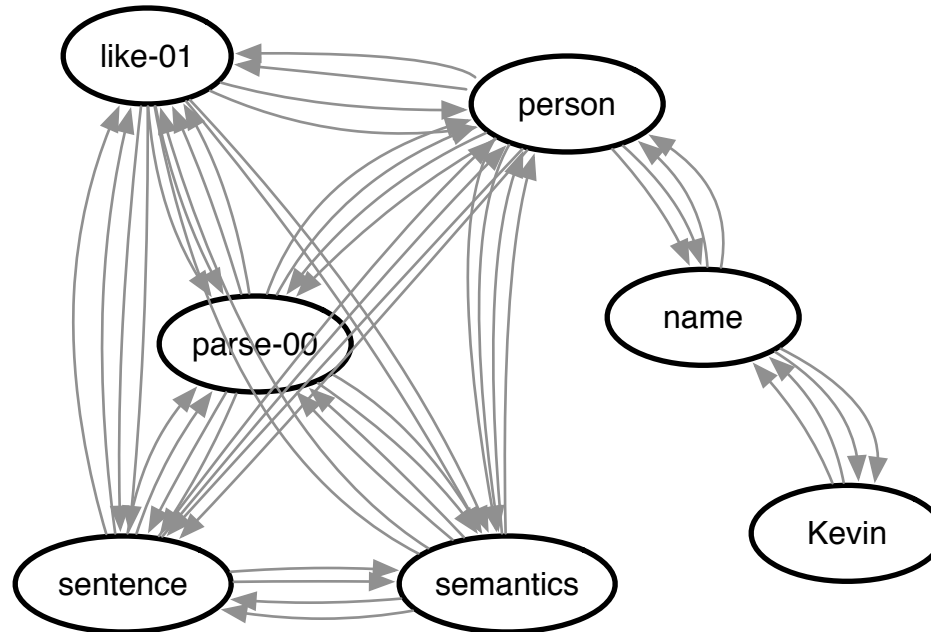




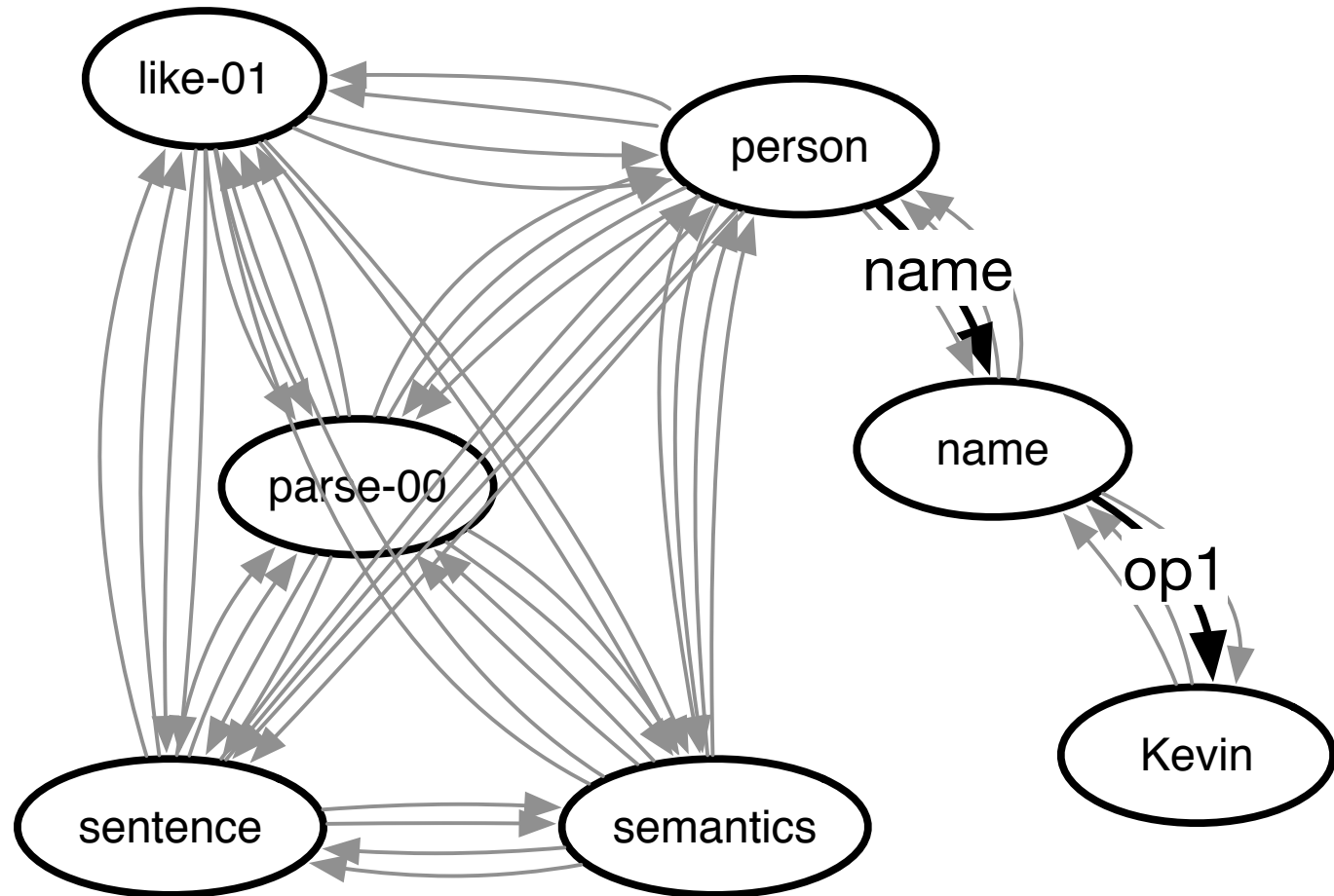
# Dense Graph



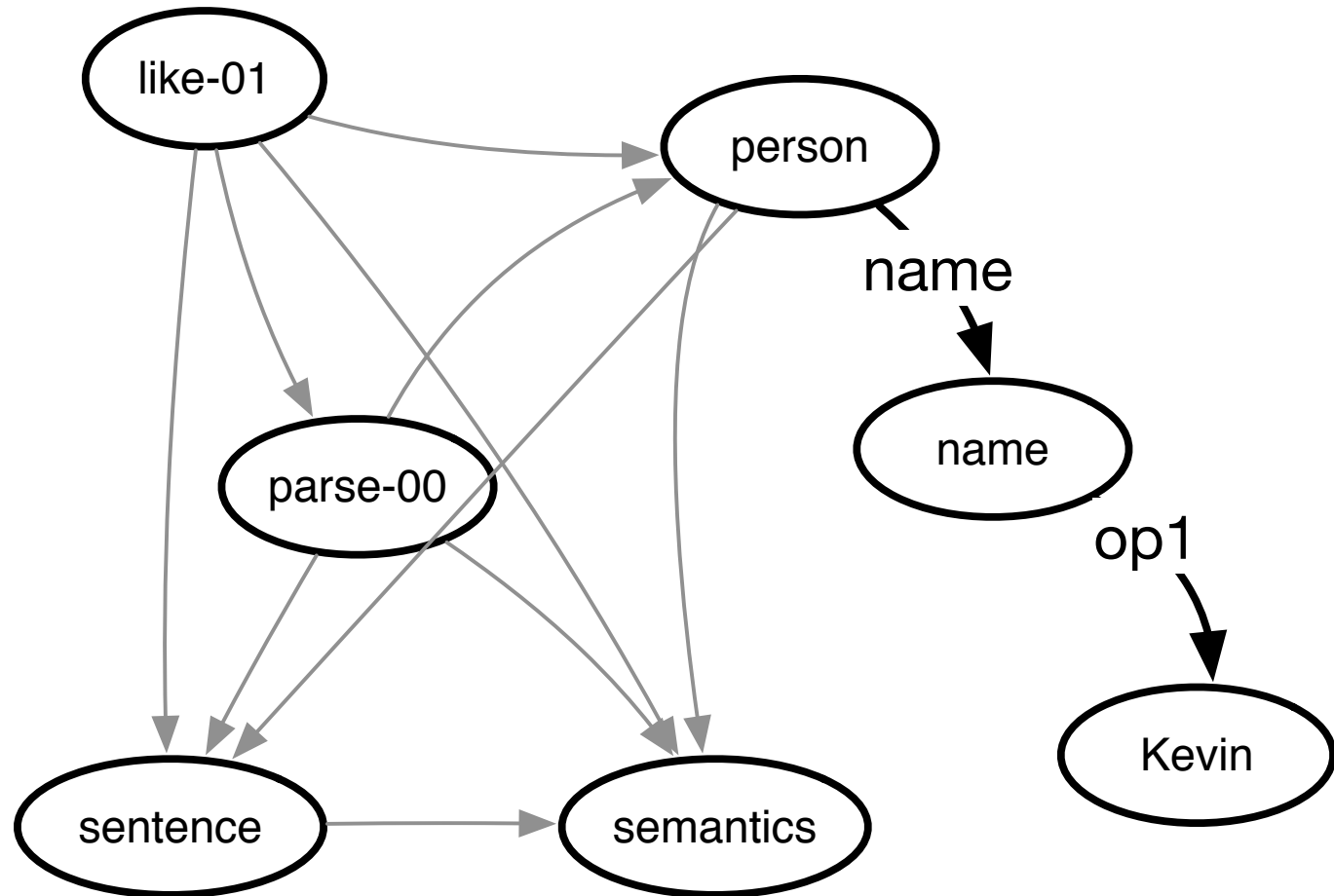
# Reduced graph for clarity



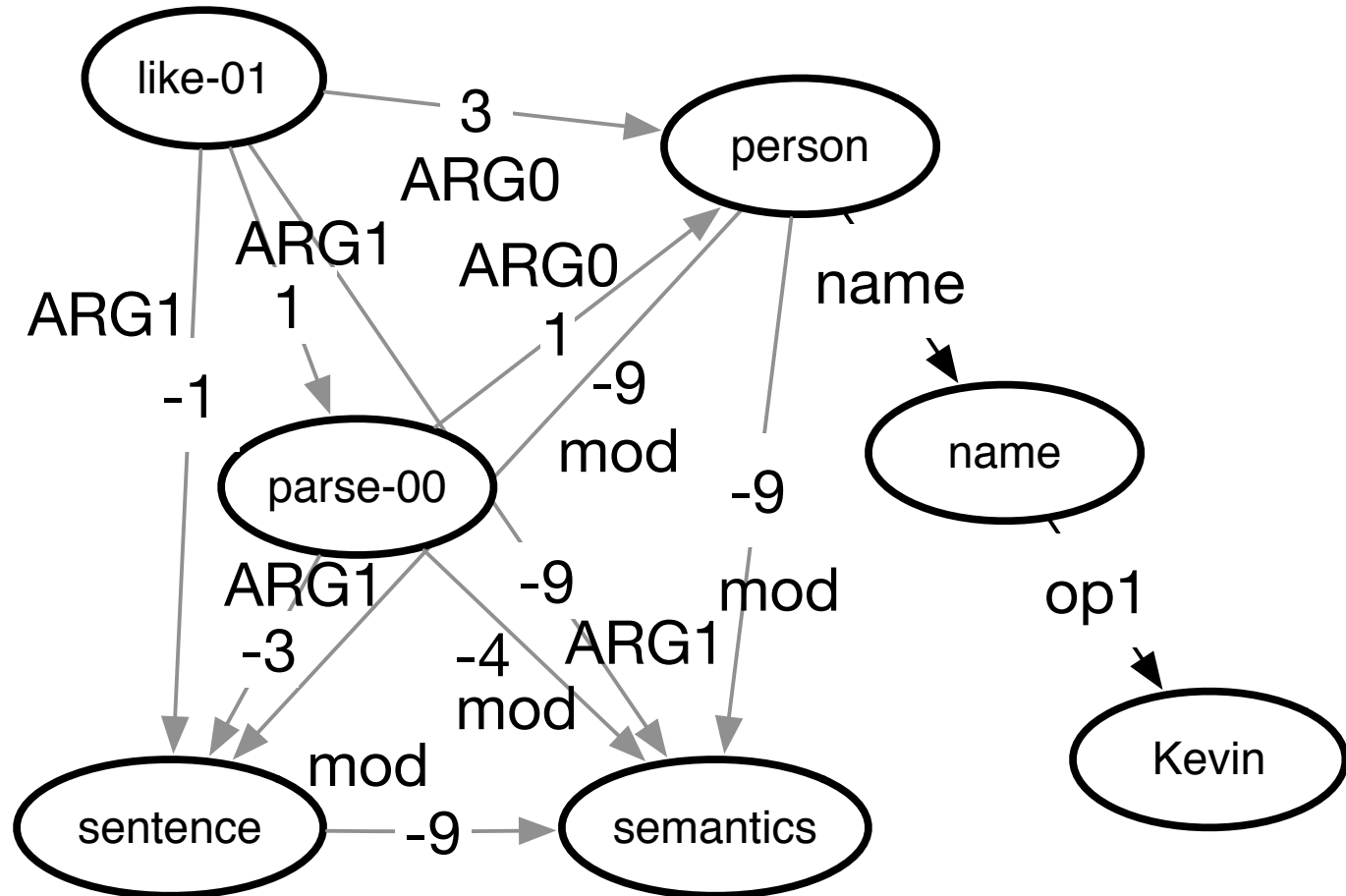
# Constraint: Preserving



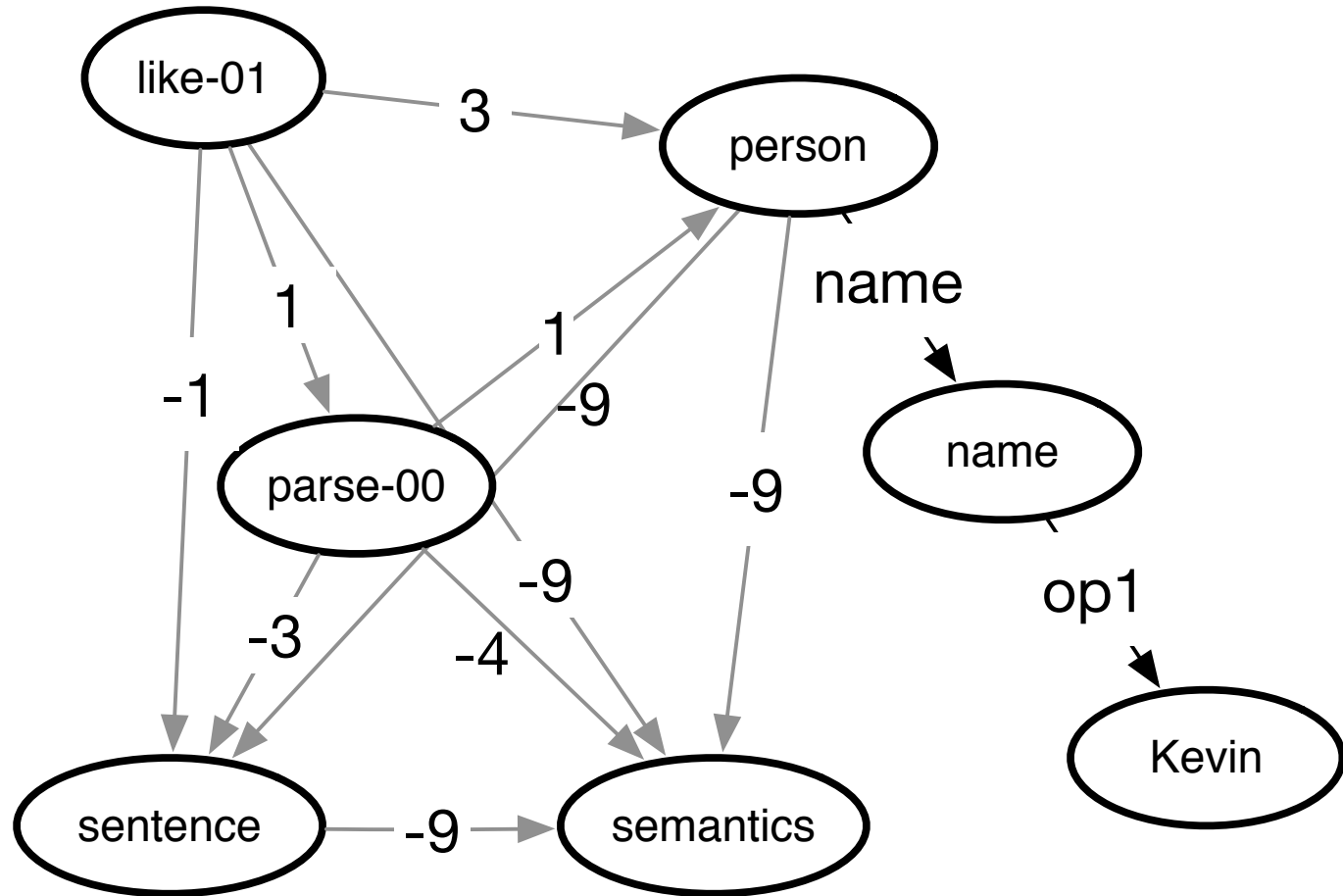
# Constraint: Simple



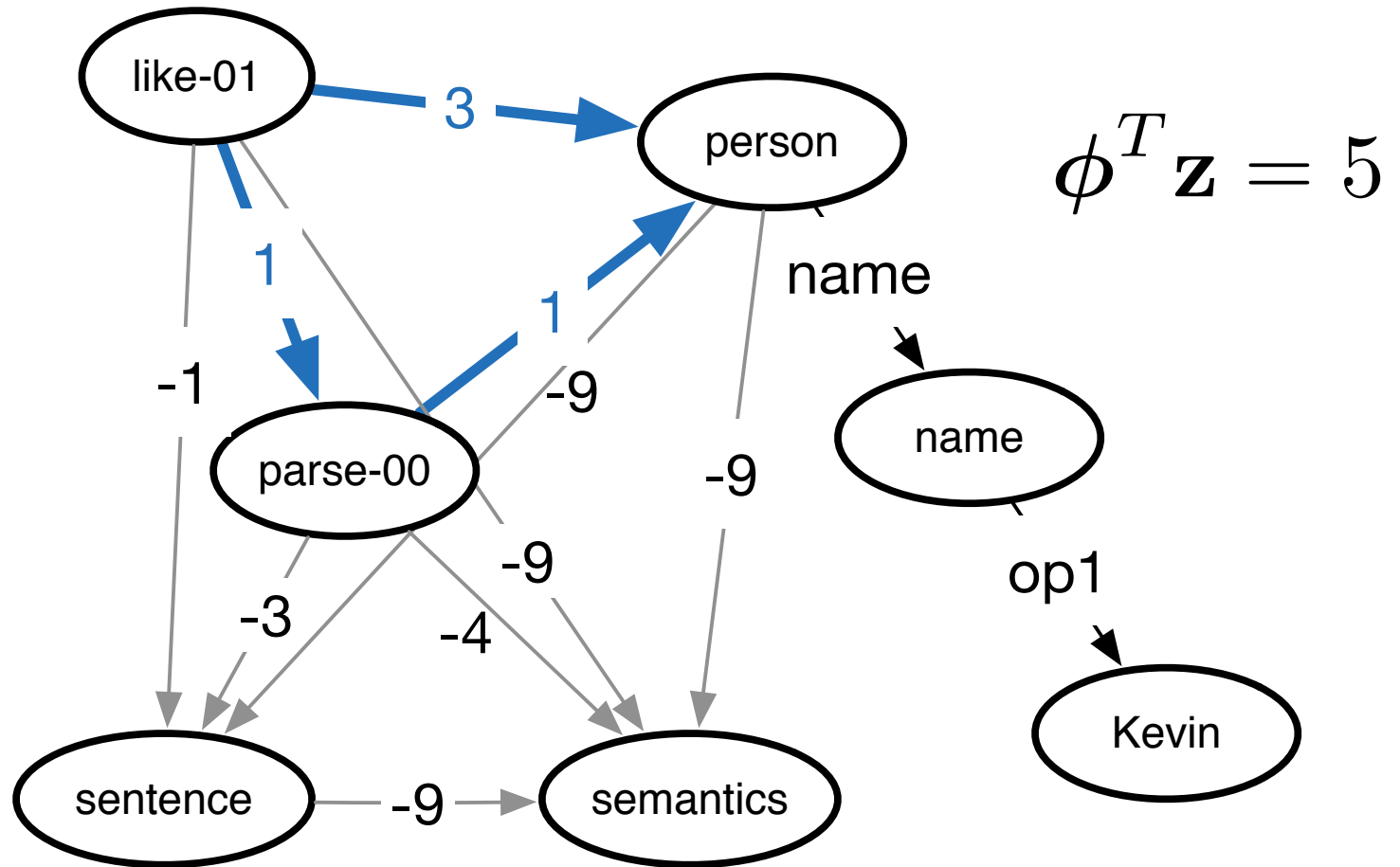
## With Weights and Labels Shown



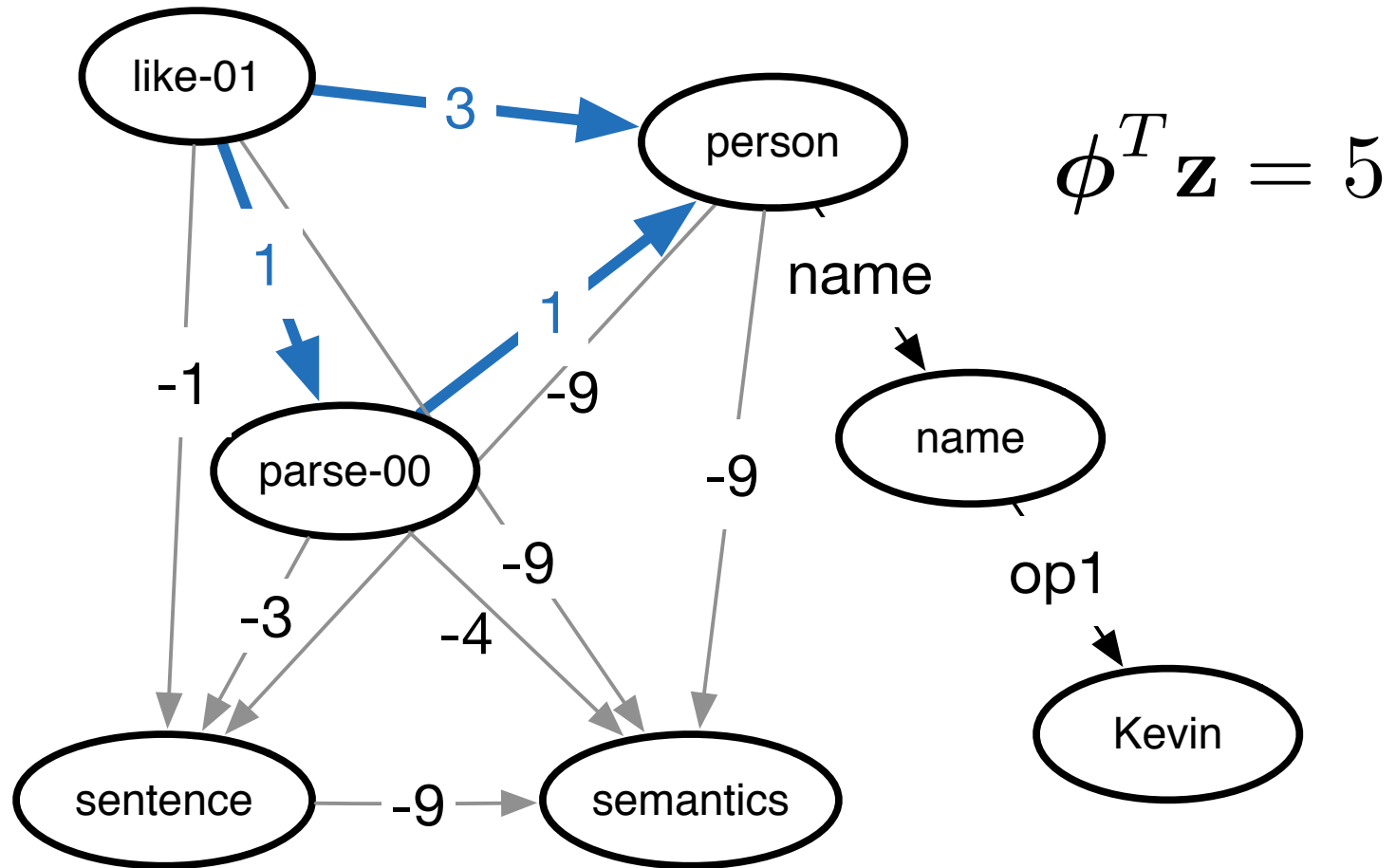
# Maximum Weighted Subgraph



# Maximum Weighted Subgraph



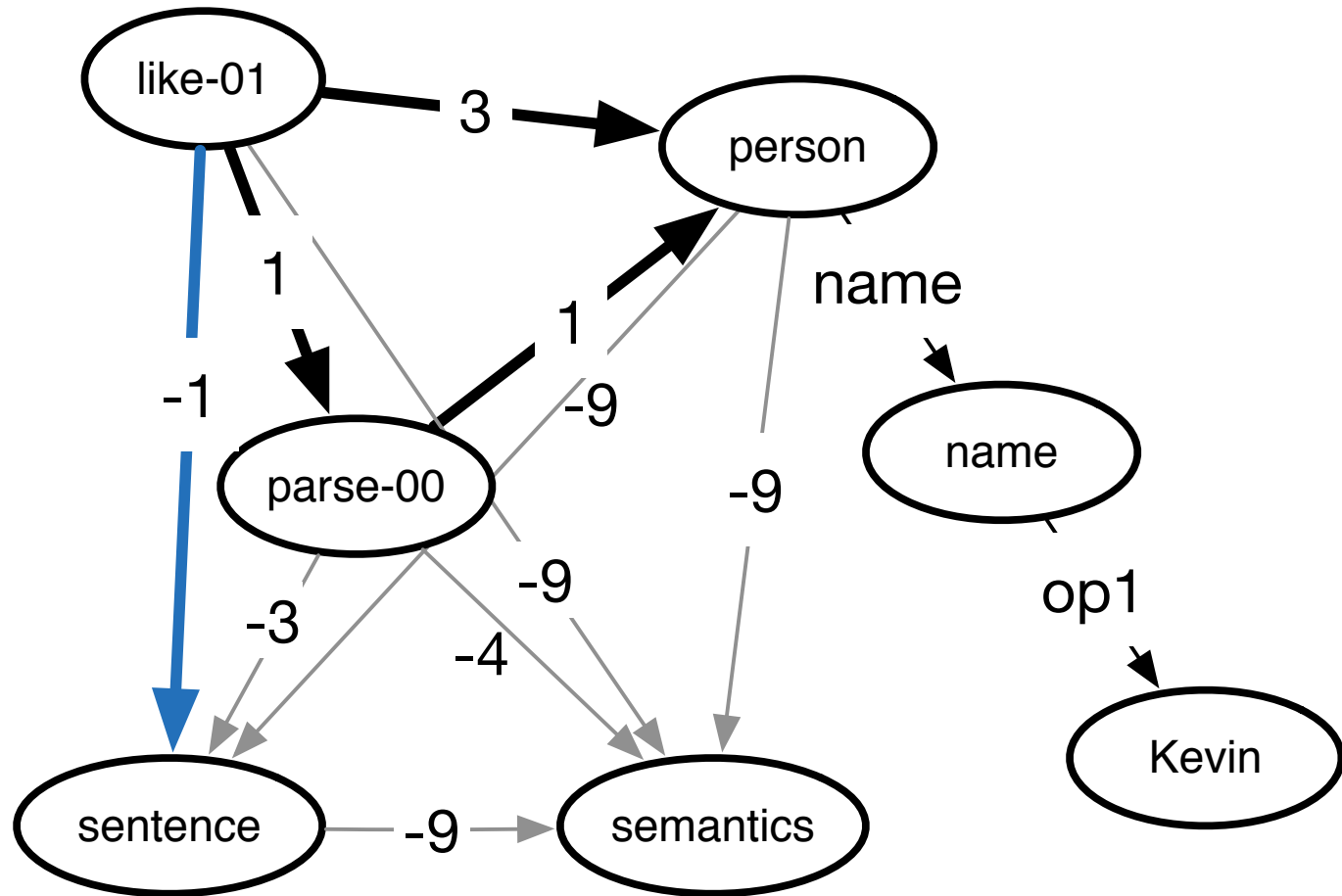
# Maximum Weighted Subgraph



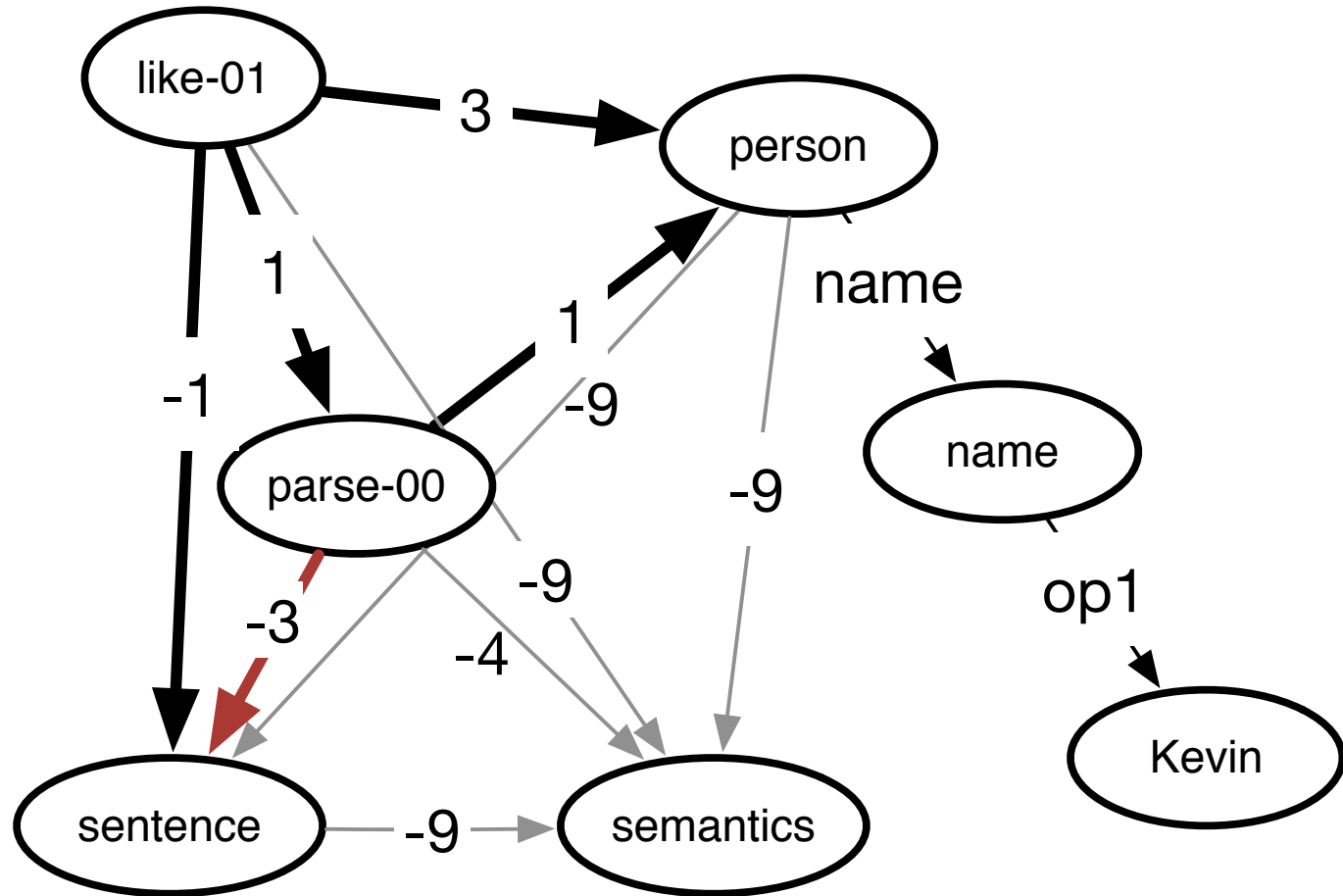
Constraint: Graph must be connected



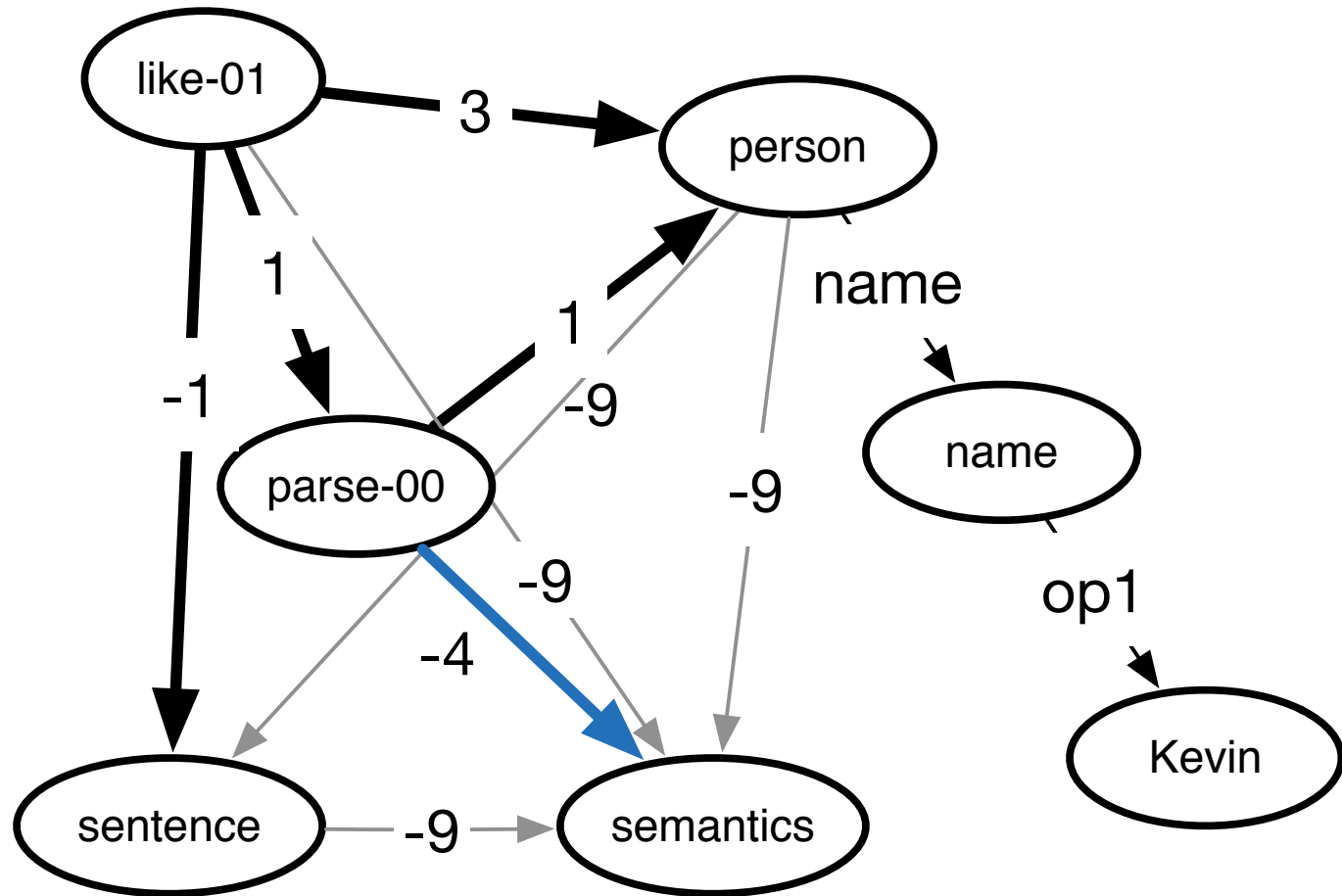
# Maximum Spanning, Connected Subgraph (MSCG)



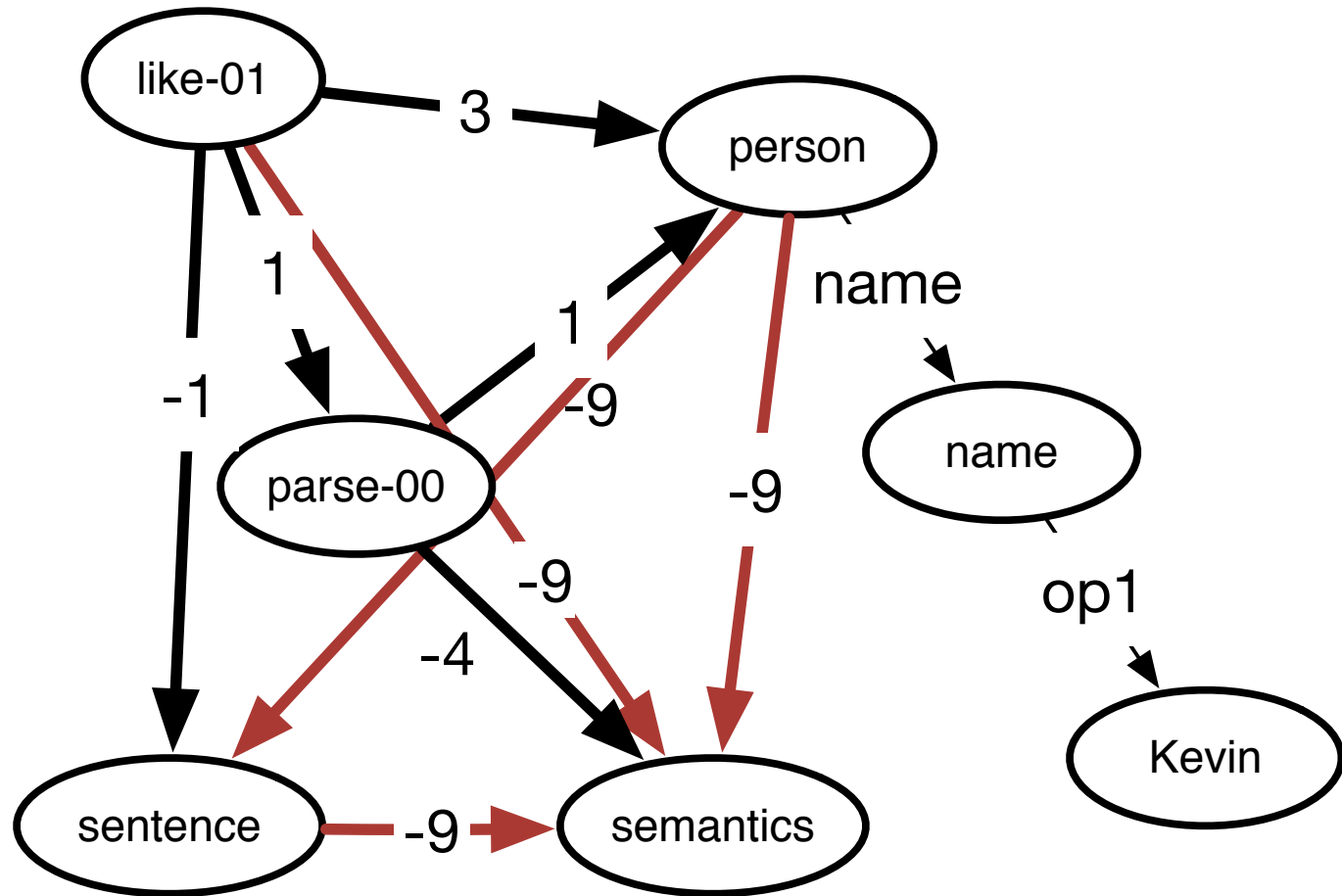
# Maximum Spanning, Connected Subgraph (MSCG)



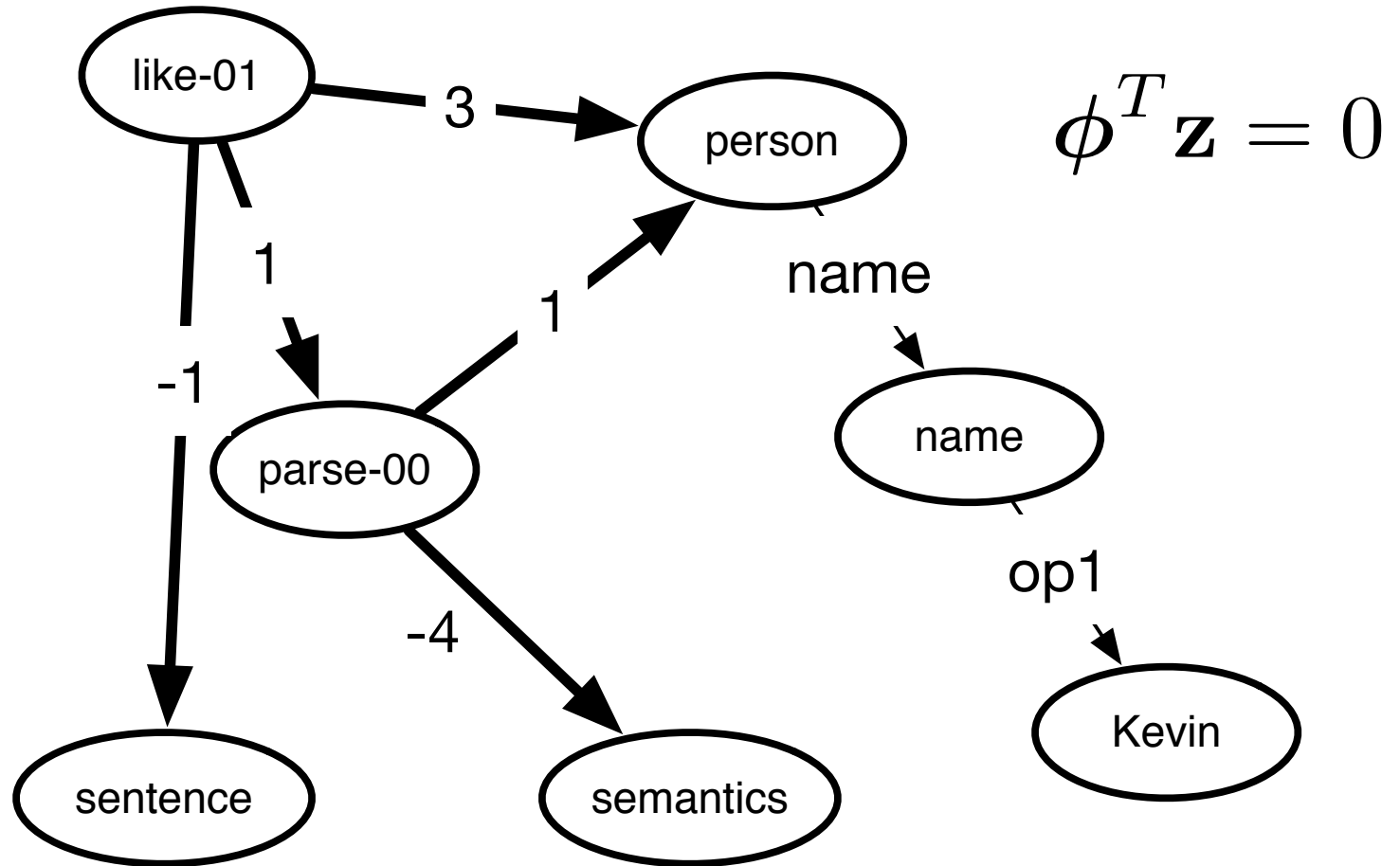
# Maximum Spanning, Connected Subgraph (MSCG)



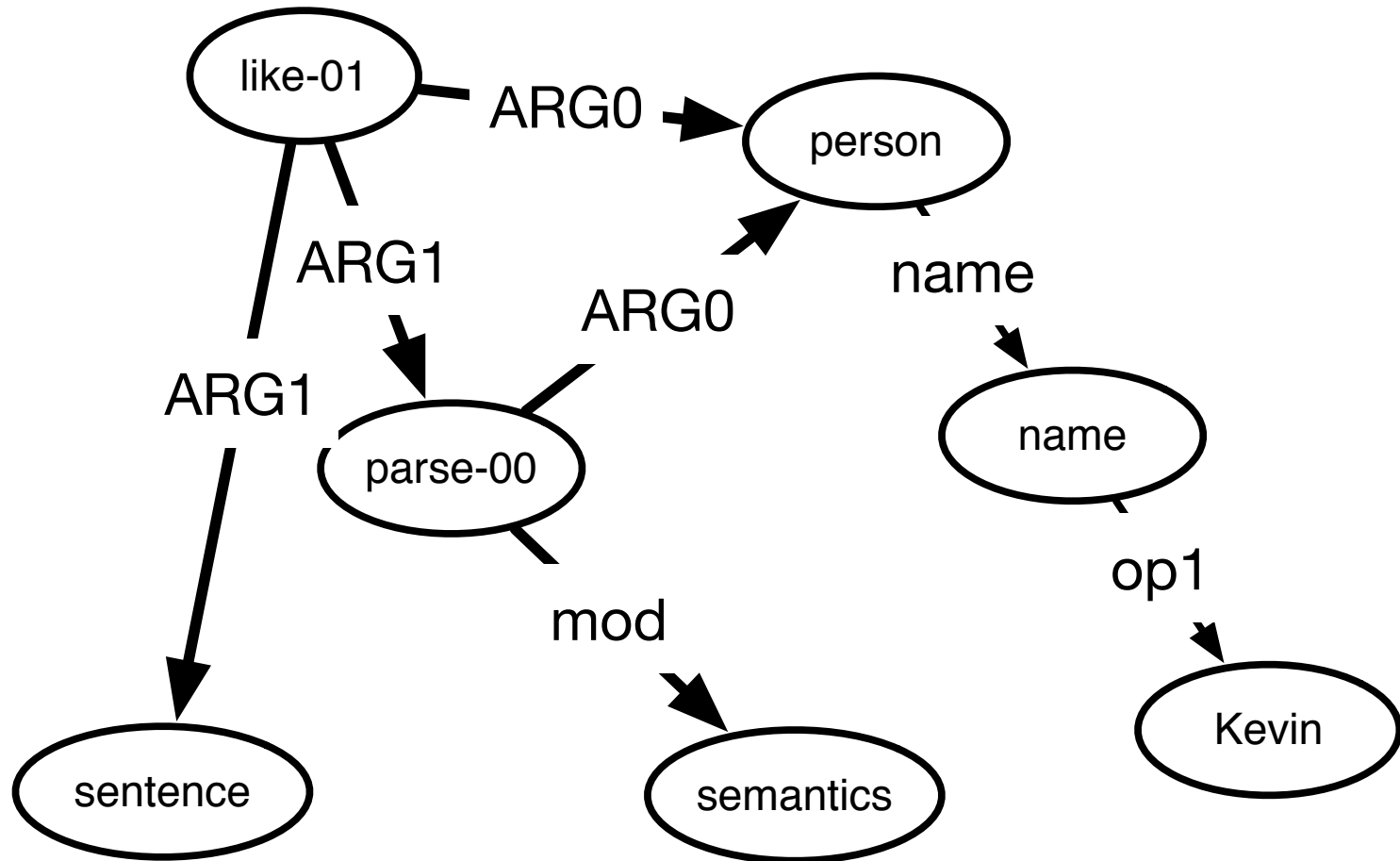
# Maximum Spanning, Connected Subgraph (MSCG)



# Maximum Spanning, Connected Subgraph (MSCG)

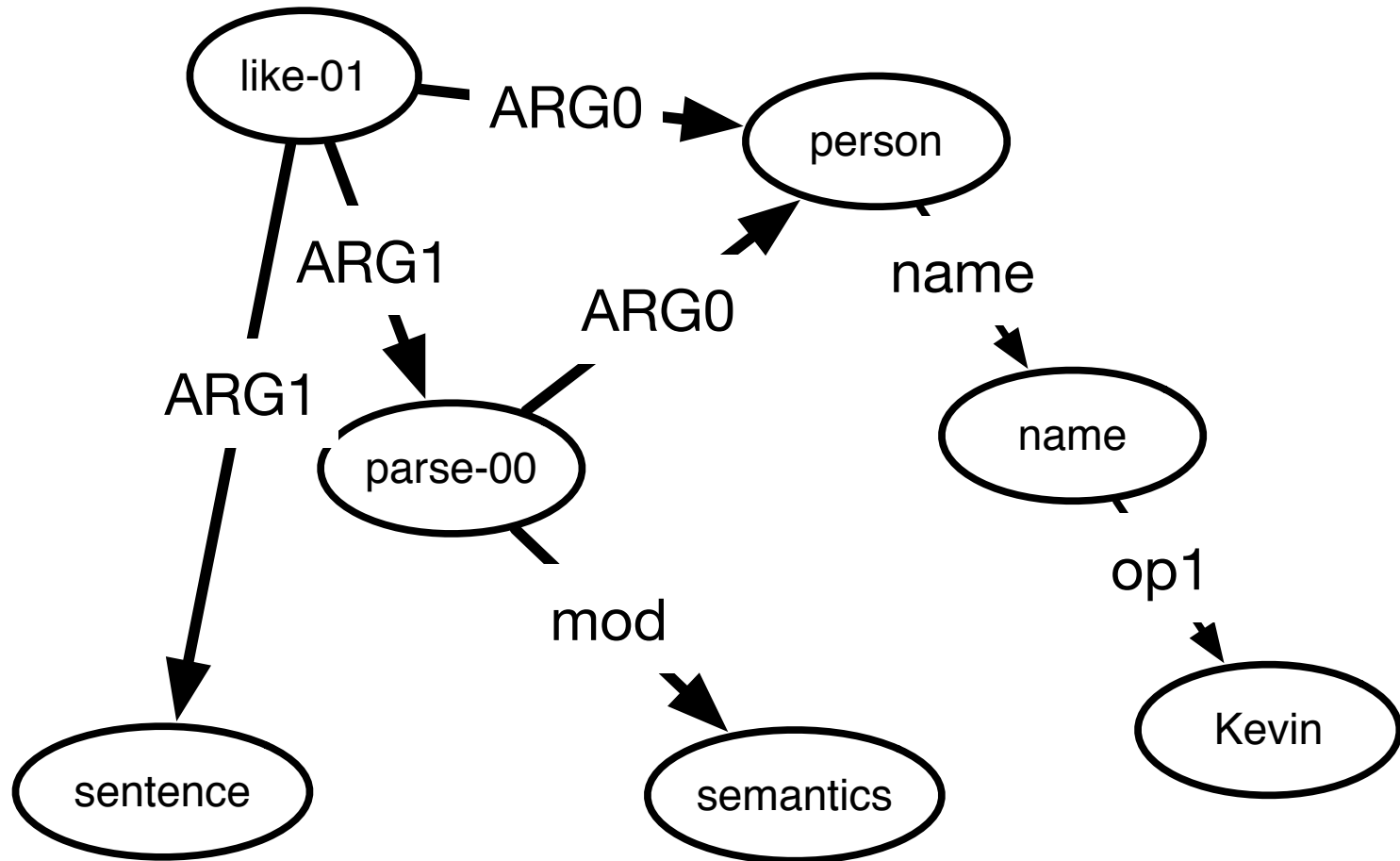


# Maximum Spanning, Connected Subgraph (MSCG)



Constraint: Graph must be deterministic

# Determinism Constraints



# Determinism Constraints

$$\mathbf{z}_{1 \xrightarrow{\text{ARG1}} 2} + \mathbf{z}_{1 \xrightarrow{\text{ARG1}} 3} + \dots \leq 1$$

$$\mathbf{z}_{2 \xrightarrow{\text{ARG1}} 1} + \mathbf{z}_{2 \xrightarrow{\text{ARG1}} 3} + \dots \leq 1$$

⋮



# Determinism Constraints

$$\mathbf{z}_{1 \xrightarrow{\text{ARG1}} 2} + \mathbf{z}_{1 \xrightarrow{\text{ARG1}} 3} + \dots \leq 1$$

$$\mathbf{z}_{2 \xrightarrow{\text{ARG1}} 1} + \mathbf{z}_{2 \xrightarrow{\text{ARG1}} 3} + \dots \leq 1$$

$\vdots$

$$A\mathbf{z} \leq b$$

# Determinism Constraints

$$\begin{array}{ll} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} & \leftarrow \text{Preserving, simple,} \\ & \text{connected, spanning} \\ \text{s.t. } \mathbf{z} \text{ satisfies } A\mathbf{z} \leq b & \leftarrow \text{Determinism} \\ & \text{constraints} \end{array}$$

Solve using Lagrangian relaxation

# Lagrangian Relaxation Tutorial

$$\max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z}$$


$$\text{s.t. } \mathbf{z} \text{ satisfies } A\mathbf{z} \leq b$$

# Lagrangian Relaxation Tutorial

$$\max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z}$$

$$\text{s.t. } \mathbf{z} \text{ satisfies } A\mathbf{z} \leq b$$

Easy (know  
how to solve)



# Lagrangian Relaxation Tutorial

$$\begin{array}{ll} \max_{\mathbf{z} \in \mathcal{Z}} & \phi^T \mathbf{z} \\ \text{s.t. } & \mathbf{z} \text{ satisfies } A\mathbf{z} \leq b \end{array}$$

Easy

But this makes it  
hard

# Lagrangian Relaxation Tutorial

$$\begin{array}{ll} \max_{\mathbf{z} \in \mathcal{Z}} & \phi^T \mathbf{z} \\ \text{s.t. } & \mathbf{z} \text{ satisfies } A\mathbf{z} \leq b \end{array}$$

Easy

Problem #1  
Hard



But this makes it  
hard

# Lagrangian Relaxation Tutorial

$$\begin{array}{ll} \max_{\mathbf{z} \in \mathcal{Z}} & \phi^T \mathbf{z} \\ \text{s.t. } & \mathbf{z} \text{ satisfies } A\mathbf{z} \leq \mathbf{b} \end{array}$$

Easy

Problem #1  
Hard



But this makes it  
hard

$$\lambda^T (\mathbf{b} - A\mathbf{z})$$

Lagrange multipliers  $\geq 0$


# Lagrangian Relaxation Tutorial

$$\max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z}$$
$$\text{s.t. } \mathbf{z} \text{ satisfies } A\mathbf{z} \leq \mathbf{b}$$

Easy

Problem #1

Hard



But this makes it hard

$$\max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

Add to original objective



# Lagrangian Relaxation Tutorial

$$\begin{array}{ll} \max_{\mathbf{z} \in \mathcal{Z}} & \phi^T \mathbf{z} \\ \text{s.t. } & \mathbf{z} \text{ satisfies } A\mathbf{z} \leq \mathbf{b} \end{array}$$

Easy

Problem #1  
Hard



But this makes it  
hard

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

Minimize over  $\lambda$

# Lagrangian Relaxation Tutorial

$$\begin{array}{ll} \max_{\mathbf{z} \in \mathcal{Z}} & \phi^T \mathbf{z} \\ \text{s.t. } & \mathbf{z} \text{ satisfies } A\mathbf{z} \leq \mathbf{b} \end{array}$$

Easy

Problem #1  
Hard



But this makes it  
hard

$\approx$

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

Problem #2  
Easy



Not always equivalent, as we shall see

## Solving Problem #2

- Problem #2 (aka “Lagrange Dual”):

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

- For a given  $\lambda$ , the max can be solved using algorithm given before (preprocessing + MSCG)
- To minimize over lambda
  - Use subgradient descent

## Solving Problem #2

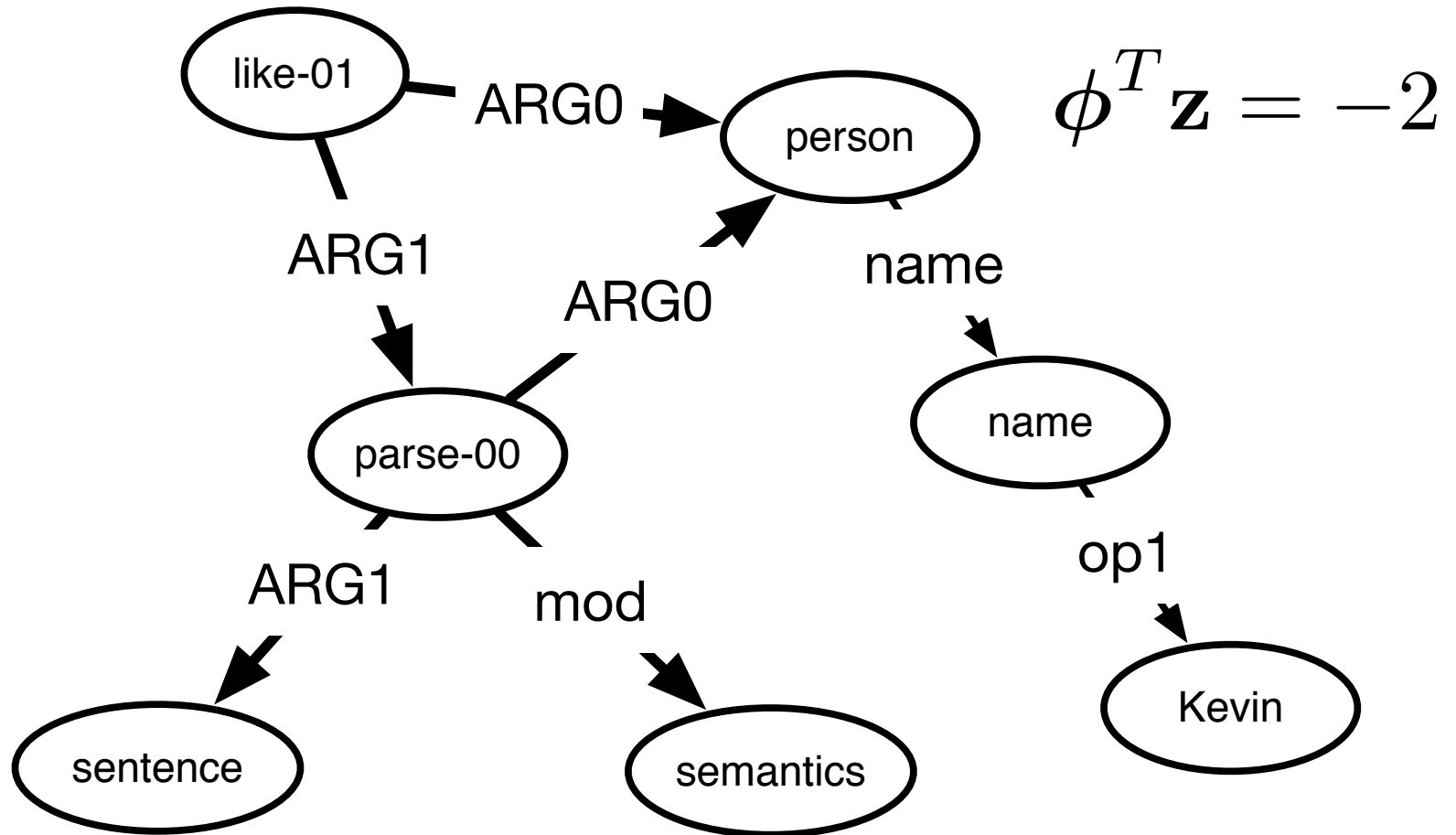
- Problem #2 (aka “Lagrange Dual”):

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

- For a given  $\lambda$ , the max can be solved using algorithm given before (preprocessing + MSCG)
- To minimize over lambda
  - Use subgradient descent

If constraints are not satisfied at minimum,  
then Problem #1  $\neq$  Problem #2

## After subgradient descent



# Summary: Output Graph Properties

- Maximum weight
- Preserving
- Simple
- Spanning (all nodes)
- Connected
- Deterministic

# Features & Training

- Features
  - Edge bias
  - Edge label
  - Head concept, tail concept, head word, tail word
  - Dependency path (dependency edge labels and POS on the shortest path between any two words in the span)
  - Various distance features
  - Within fragment edge indicator
  - Various conjunctions of above features
- Weights trained using AdaGrad structured perceptron

# Experiments

- Alignment
- Parsing
  - Graph-based parsing
    - Concept identification
    - Relation identification
      - Maximum spanning connected graph algorithm (MSCG)
      - Graph determinism constraints using Lagrangian relaxation
  - **Experiments**
    - Transition-based parsing
    - Parsing using syntax-based MT
- Evaluation
- Graph Grammars and Automata
- Applications



# Experiments

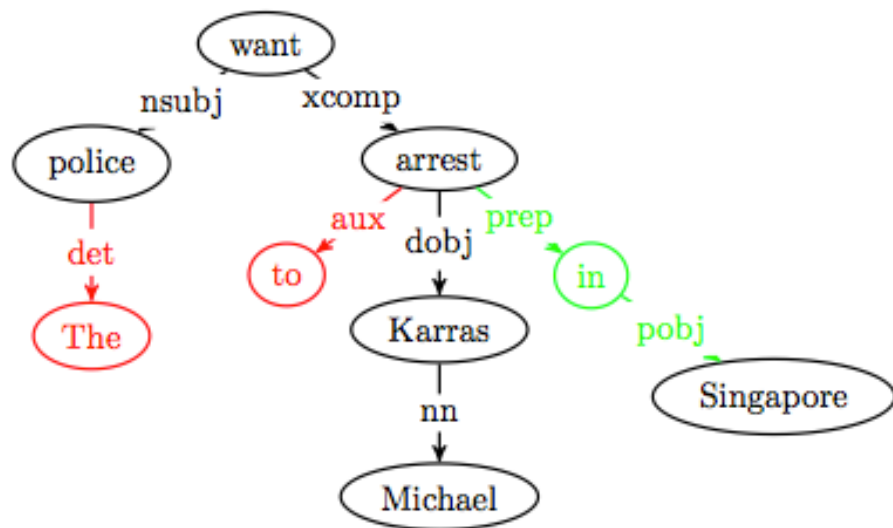
	<b>ACL 2014</b>	<b>Now</b>
Full System (gold concepts)	80% Smatch	81% Smatch
Full System	58% Smatch	62% Smatch

- Data: LDC2013E117
  - 4,000 training instances
  - 2,000 test
  - 2,000 dev

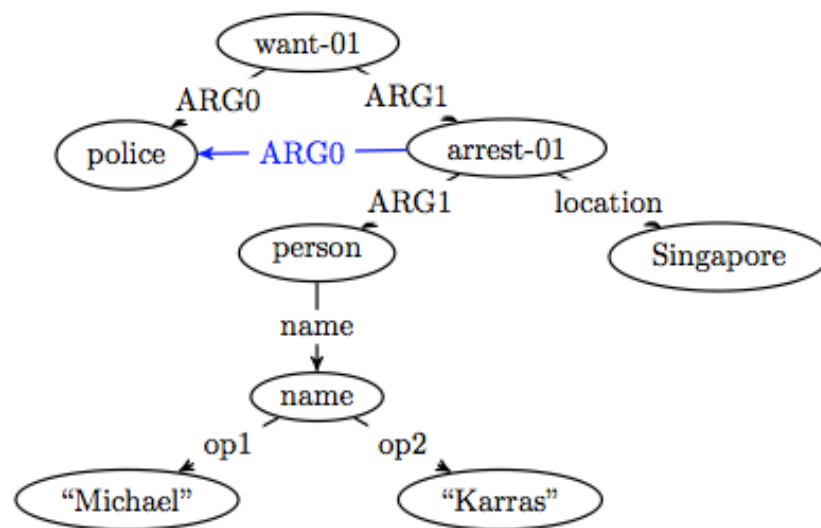
# Transition-based AMR Parsing (Wang et al, NAACL 2015)

- Convert dependency tree into AMR graph
- Motivation: only a few difference between syntactic dependencies and AMR

Dependency tree



AMR graph

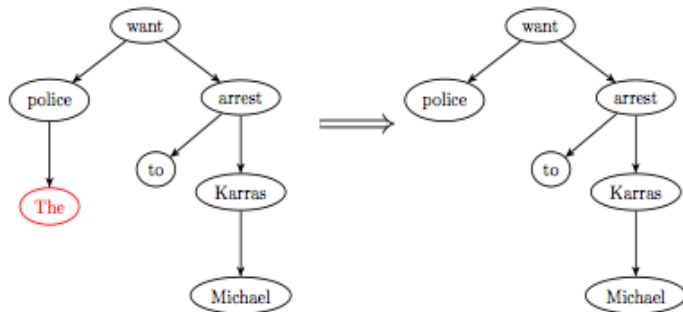


# Transition-based AMR Parsing

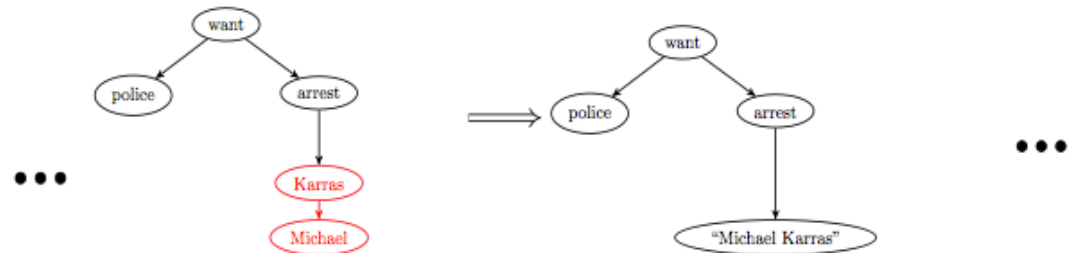
- Actions applied to graph in post-order traversal
- Parser actions
  - **NEXT-EDGE- $I_r$**  (attach edge and move to next word)
  - **SWAP- $I_r$**  (swap nodes and attach with edge)
  - **REATTACH $_k$ - $I_r$**  (delete edge and reattach to already processed node)
  - **REPLACE-HEAD** (replace node with another node)
  - **REENTRANCE $_k$ - $I_r$**  (attach edge to already processed node)
  - **MERGE** (merge two nodes)
  - **NEXT-NODE- $I_c$**  (label with concept and move to next word)
  - **DELETE-NODE** (deletes a word)

# Transition-based AMR Parsing

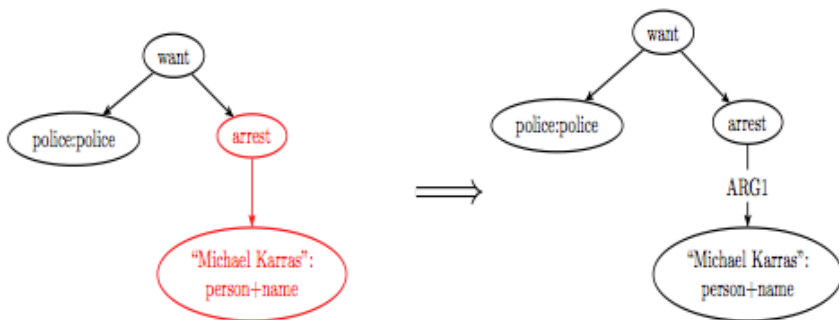
## DELETE-NODE



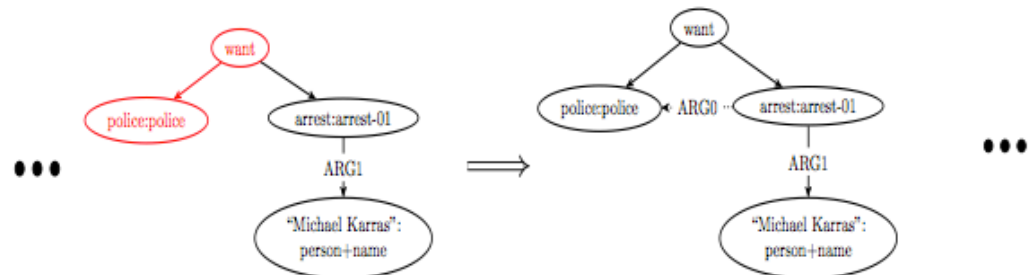
## MERGE



## NEXT-EDGE-ARG1



## REENTRANCE<sub>arrest</sub>-ARG0



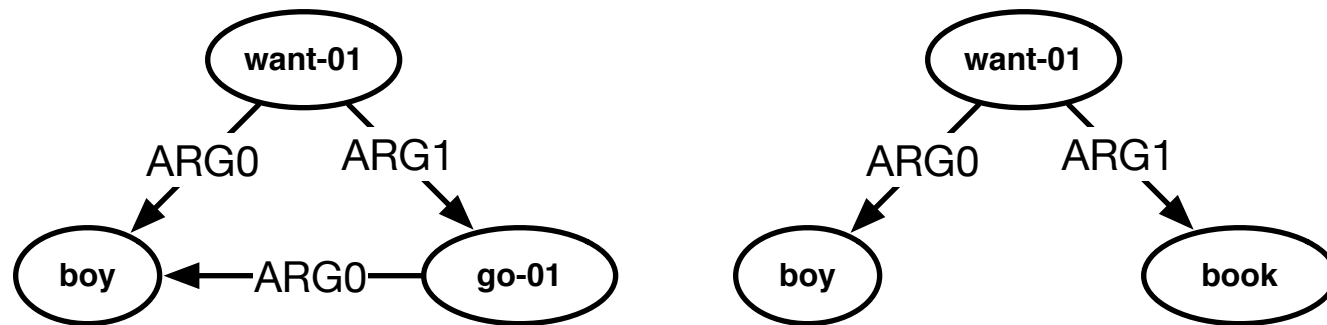
# AMR Parsing using Syntax-based MT (Pust et al, 2015)

- Idea: already have sophisticated string-to-tree syntactic MT systems. Use them for AMR parsing
- Approach: convert AMR graphs into trees suitable for training string-to-tree MT systems
- Important features:
  - Language model on the linearized AMR
  - Semantic categories built using WordNet
- Large performance gains JAMR (7 smatch points)

# Evaluation

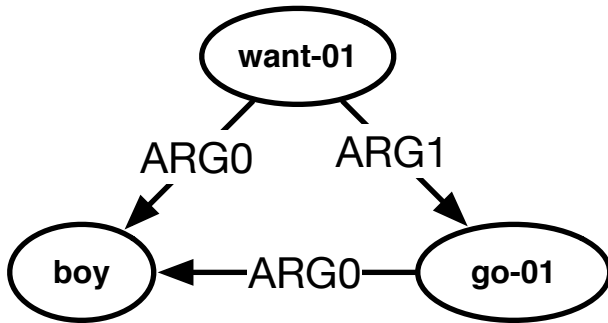
- Alignment
- Parsing
- **Evaluation**
- Graph Grammars and Automata
- Applications

# Evaluation: Smatch (Cai & Knight, 2013)

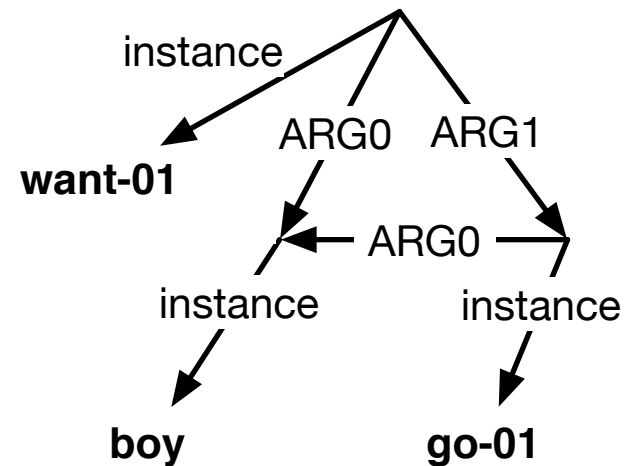


Want a number which indicates the similarity between two graphs

# Evaluation: Smatch



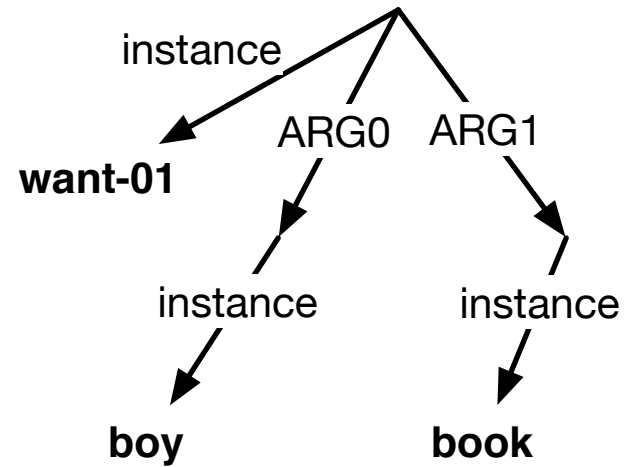
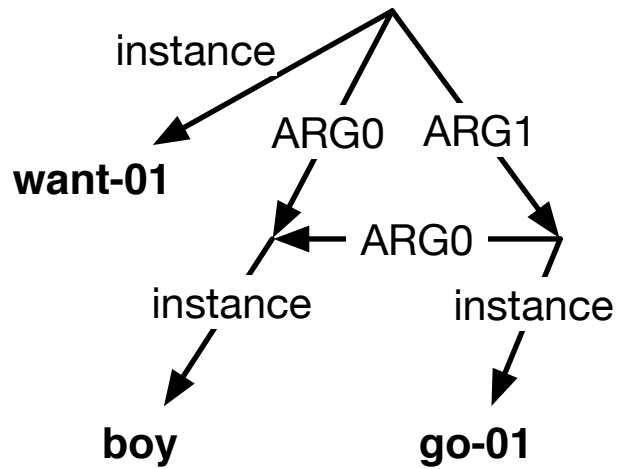
```
(a / want-01
  :ARG0 (b / boy
    :ARG1 (c / go-01
      :ARG0 b) )
```



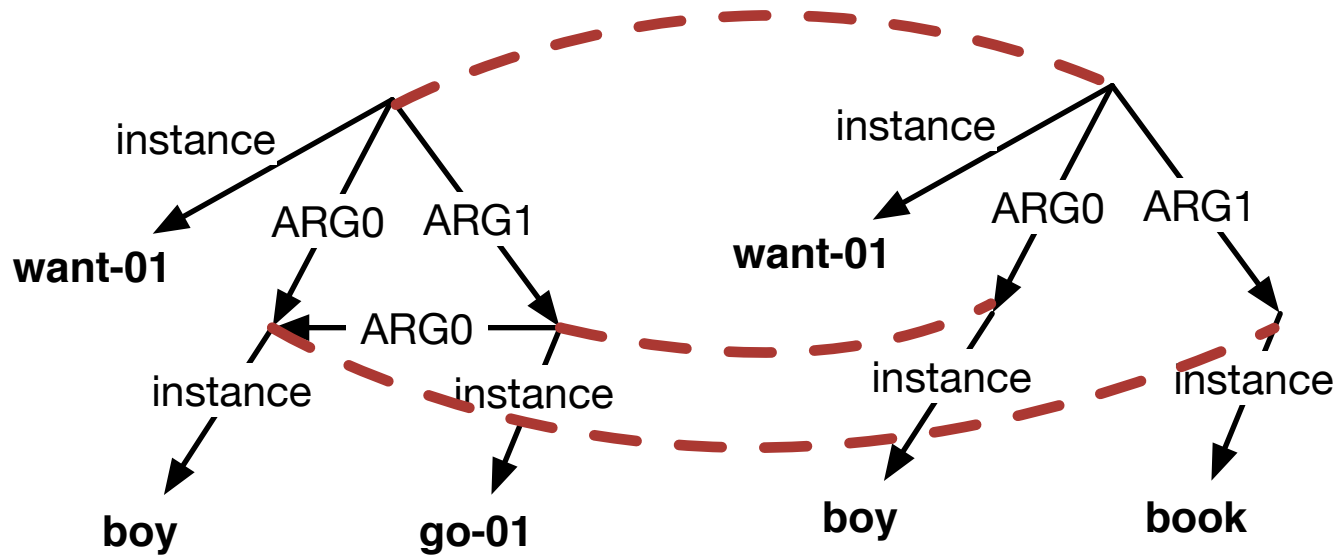
AMR graph w/ explicit instance edges



# Evaluation: Smatch

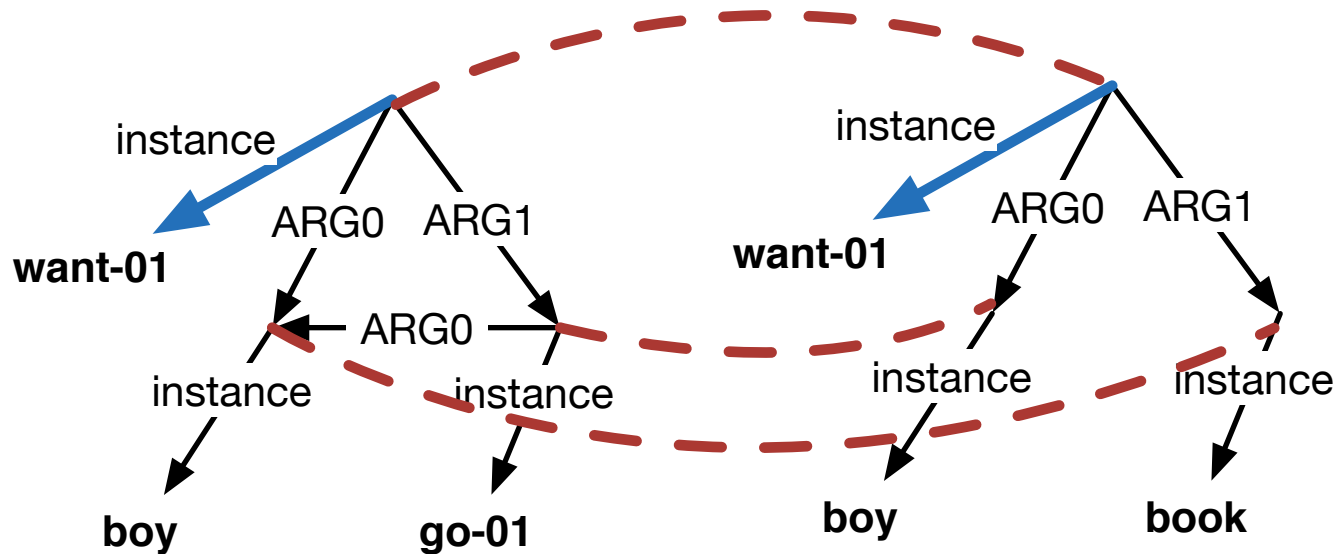


# Evaluation: Smatch



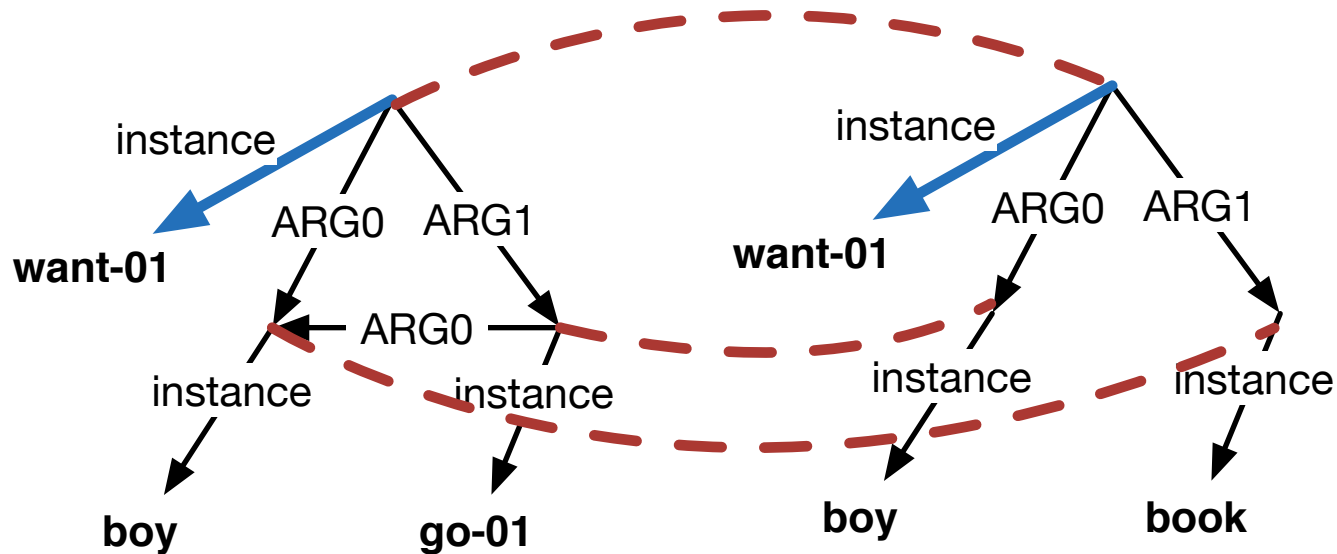
Consider an alignment between the nodes

# Evaluation: Smatch



$$\begin{aligned}\text{f-score} &= F_1 \text{ of identical matching edges} \\ &= 2 \text{ Match} / (\text{Total}_1 + \text{Total}_2) \\ &= 2 / (6 + 5) = .18\end{aligned}$$

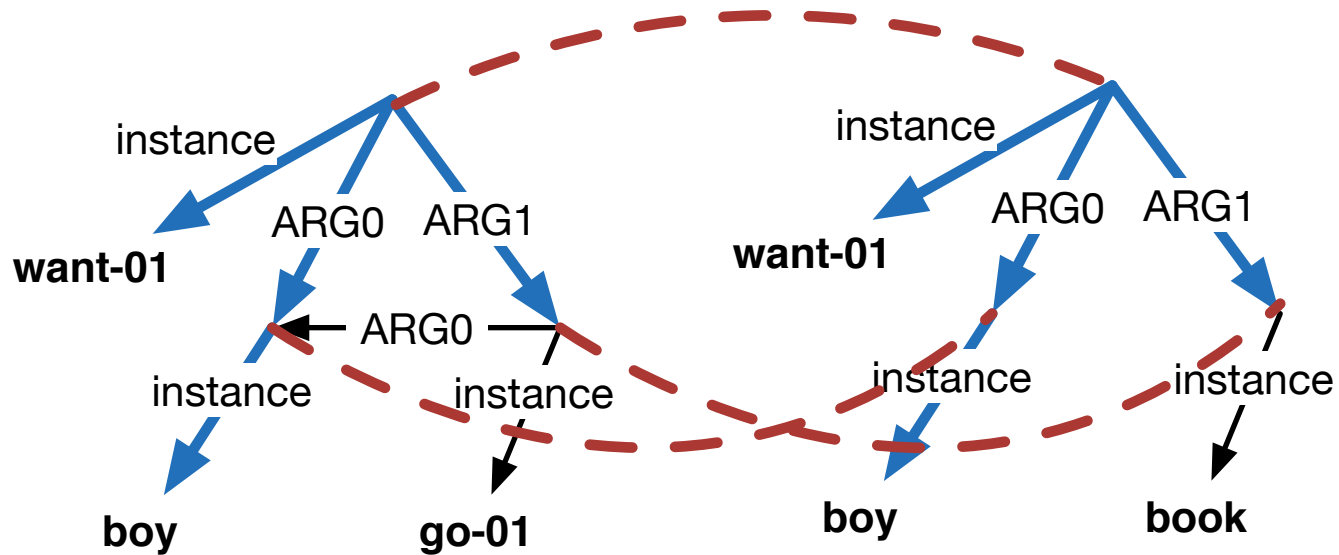
# Evaluation: Smatch



Smatch score = maximum f-score over all possible alignments

NP hard => approximate inference to find highest scoring alignment

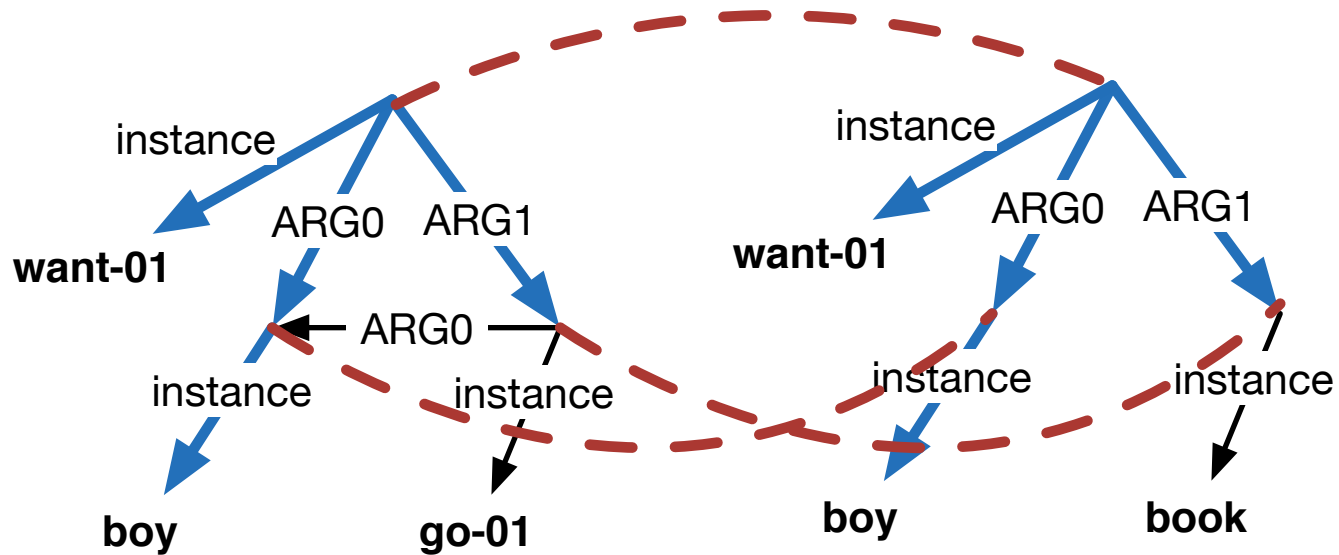
# Evaluation: Smatch



$$\text{Smatch score} = 8 / (6 + 5) = .73$$

Highest scoring alignment

# Evaluation: Smatch



Multi-lingual version of Smatch: AMRICA  
demo by Naomi Saphra at NAACL 2015

# Roadmap

- Alignment
- Parsing
- Evaluation
- **Graph Grammars and Automata**
  - Background: CFGs and tree substitution grammars
  - Hyperedge Replacement Grammars (HRGs)
  - Directed Acyclic Graph (DAG) Automata
- Applications

# Motivation for Graph Grammars

- String and tree grammars, automata, transducers, etc widely used in NLP applications
  - Phrase structure parsers, syntactic MT systems
- Semantics (like AMR) is represented as graphs

We would like grammars, automata, transducers, etc over graphs



# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

## Example derivation

S

# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

## Example derivation

$S$   
 $\Rightarrow_1 NP VP$

# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

## Example derivation

$S$   
 $\Rightarrow_1 NP VP$   
 $\Rightarrow_3 NP V NP$

# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

## Example derivation

$S$   
 $\Rightarrow_1 NP VP$   
 $\Rightarrow_3 NP V NP$   
 $\Rightarrow_4 NP like NP$

# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

## Example derivation

$S$   
 $\Rightarrow_1 NP VP$   
 $\Rightarrow_3 NP V NP$   
 $\Rightarrow_4 NP like NP$   
 $\Rightarrow_6 NP like ice\ cream$

# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

## Example derivation

$S$   
 $\Rightarrow_1 NP VP$   
 $\Rightarrow_3 NP V NP$   
 $\Rightarrow_4 NP like NP$   
 $\Rightarrow_6 NP like ice\ cream$   
 $\Rightarrow_2 We like ice\ cream$

# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

Derivation

## Example derivation

S

$\Rightarrow_1 NP VP$

$\Rightarrow_3 NP V NP$

$\Rightarrow_4 NP like NP$

$\Rightarrow_6 NP like ice cream$

$\Rightarrow_2 We like ice cream$



# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

Derivation

## Example derivation

$S$   
 $\Rightarrow_1 NP VP$   
 $\Rightarrow_3 NP V NP$   
 $\Rightarrow_4 NP like NP$   
 $\Rightarrow_6 NP like ice\ cream$   
 $\Rightarrow_2 We like ice\ cream$

Yield = **string**

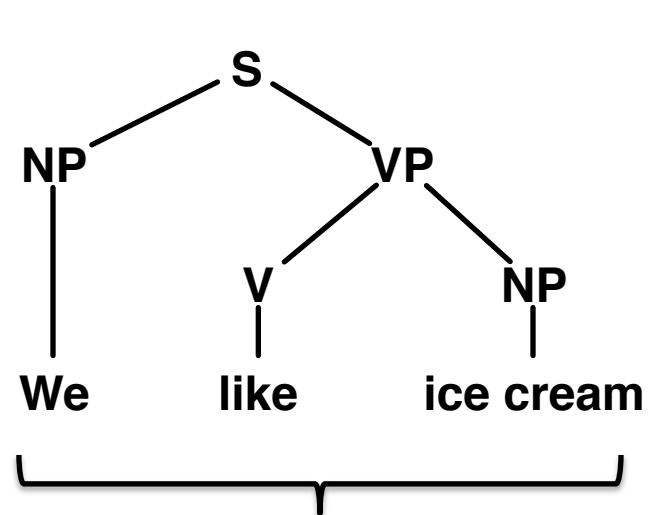
# Context Free Grammar (CFG)

## Grammar

- 1)  $S \rightarrow NP VP$
- 2)  $NP \rightarrow We$
- 3)  $VP \rightarrow V NP$
- 4)  $V \rightarrow want$
- 5)  $V \rightarrow like$
- 6)  $NP \rightarrow ice\ cream$

Derivation  
tree

## Example derivation

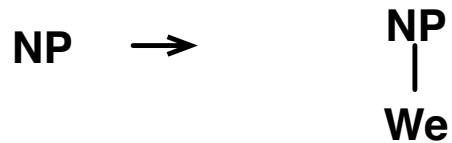
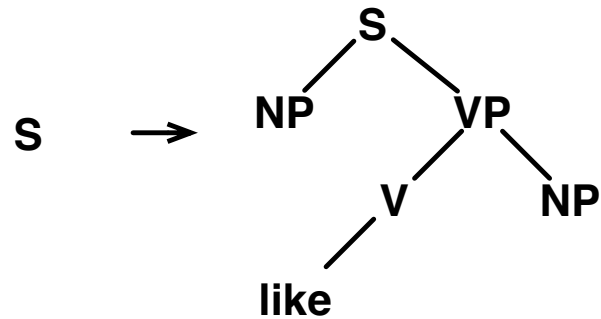


Yield = **string**

Language over strings (yield), and  
trees (derivations)

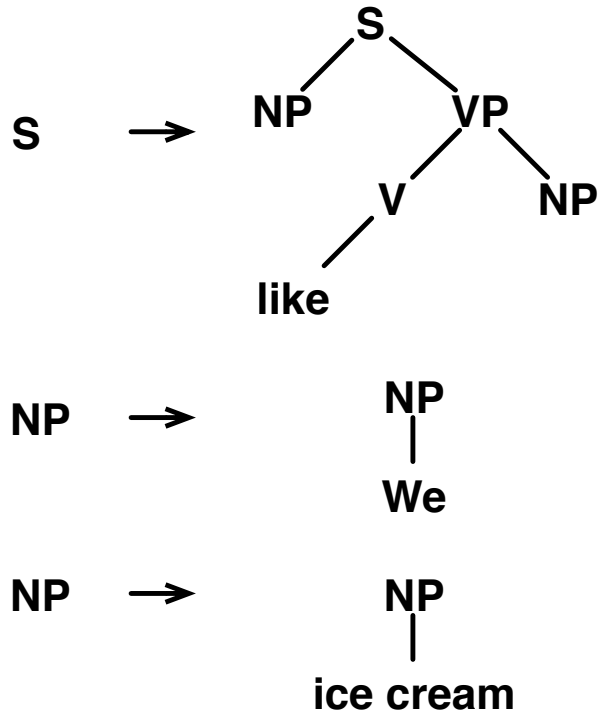
# Tree Substitution Grammar (TSG)

## Grammar



# Tree Substitution Grammar (TSG)

## Grammar

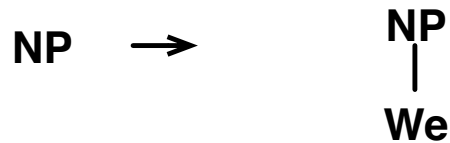
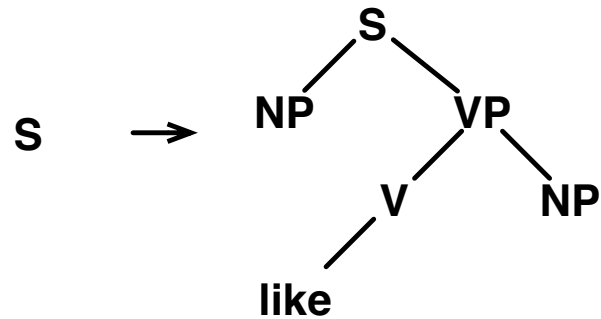


## Example derivation

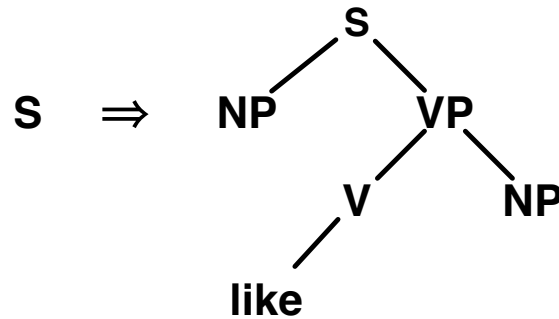
S

# Tree Substitution Grammar (TSG)

## Grammar

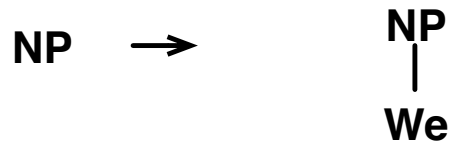
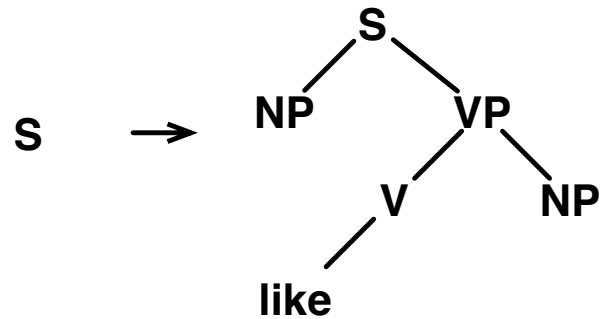


## Example derivation

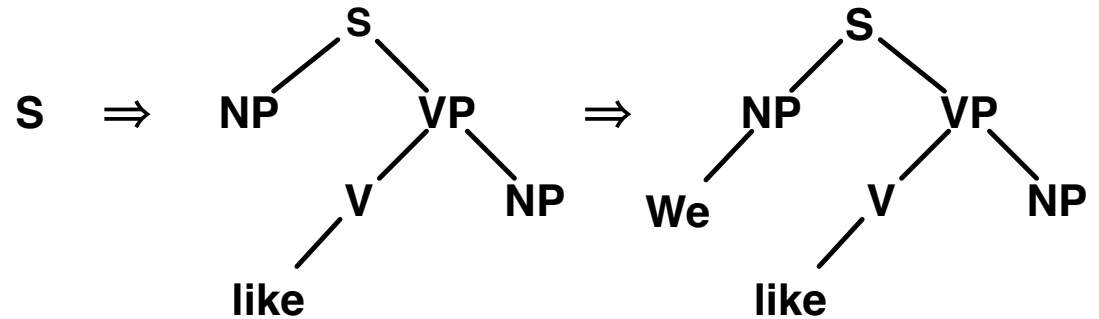


# Tree Substitution Grammar (TSG)

## Grammar

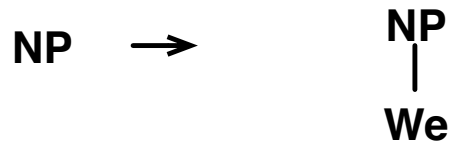
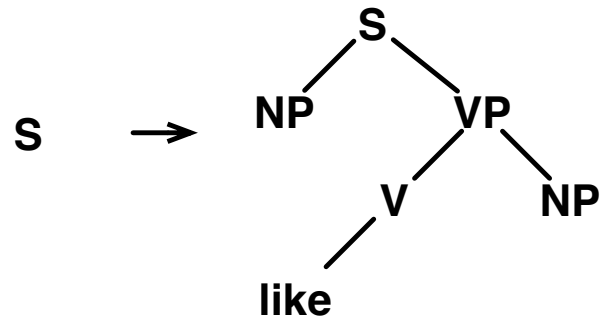


## Example derivation

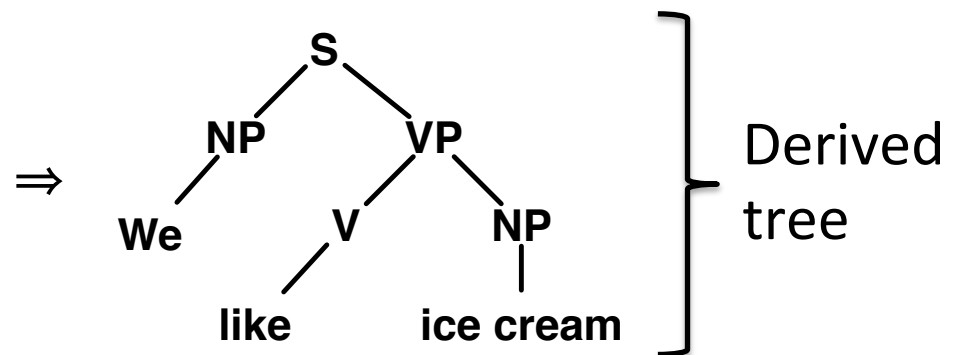
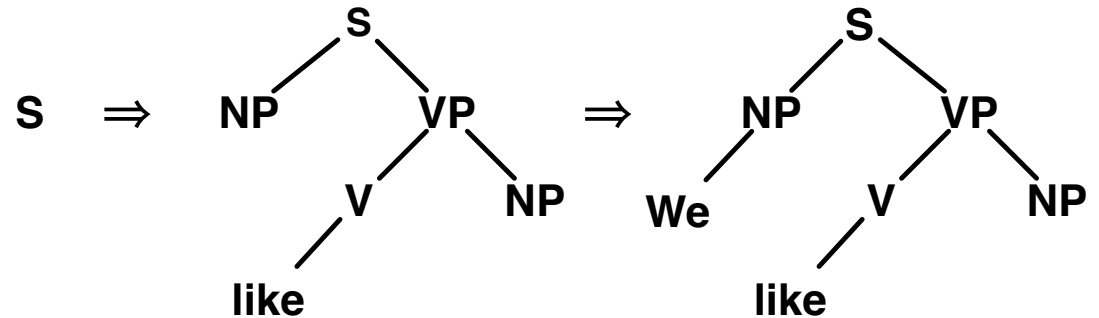


# Tree Substitution Grammar (TSG)

## Grammar

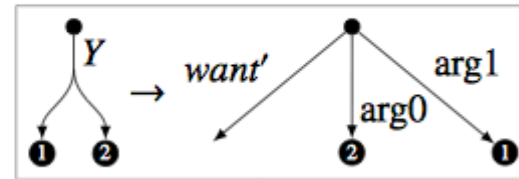
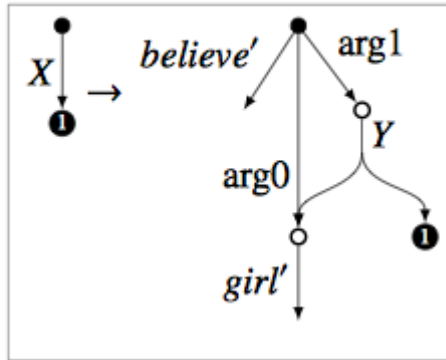


## Example derivation



# Hyperedge Replacement Grammar (HRG)

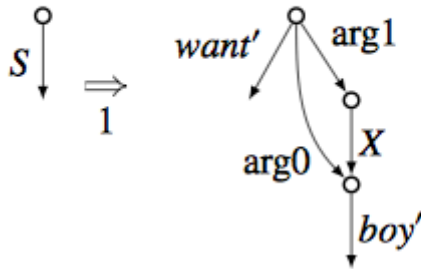
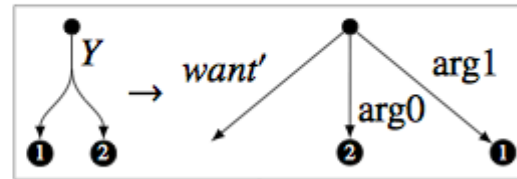
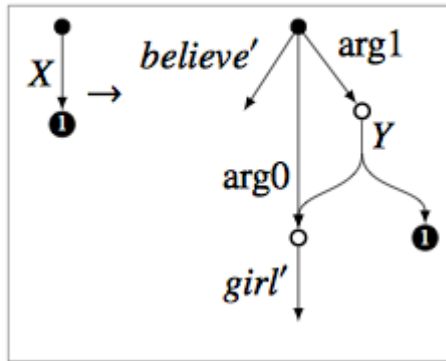
## Grammar rules (some of them)





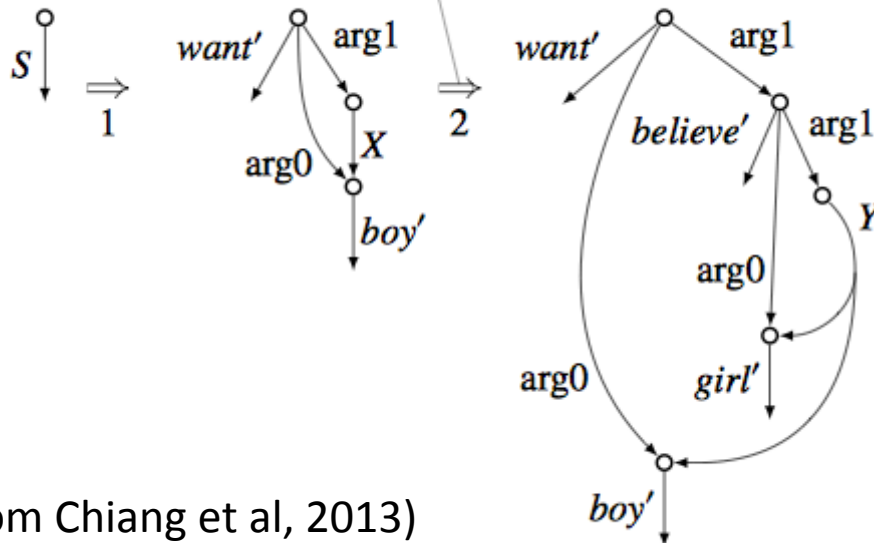
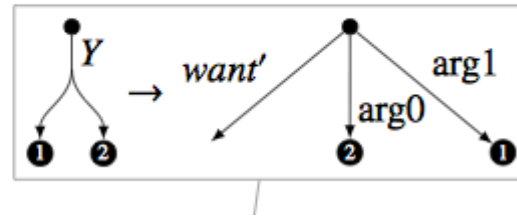
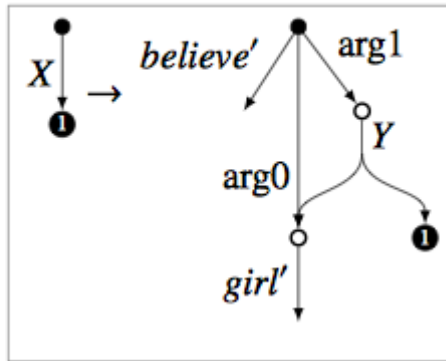
# Hyperedge Replacement Grammar (HRG)

## Grammar rules (some of them)



# Hyperedge Replacement Grammar (HRG)

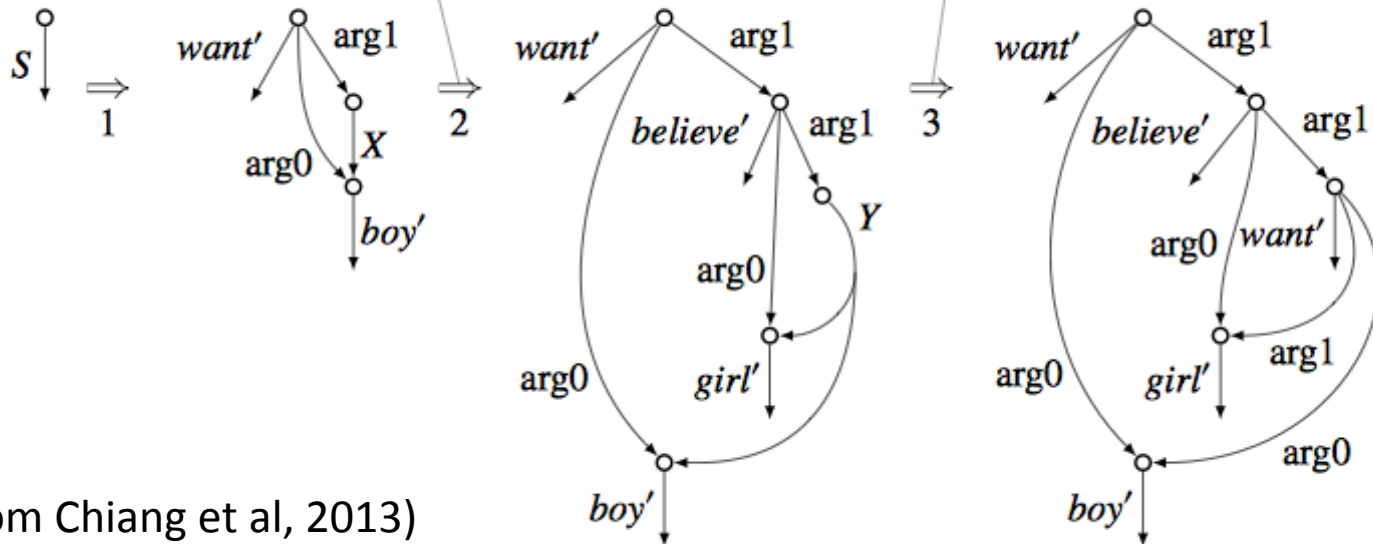
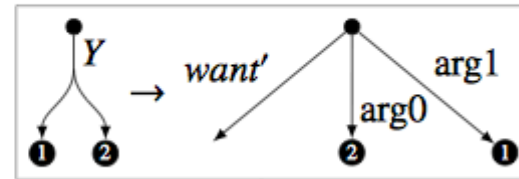
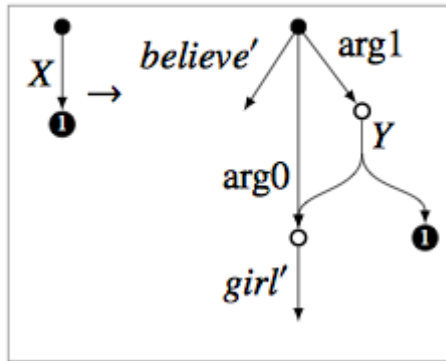
## Grammar rules (some of them)



(figure from Chiang et al, 2013)

# Hyperedge Replacement Grammar (HRG)

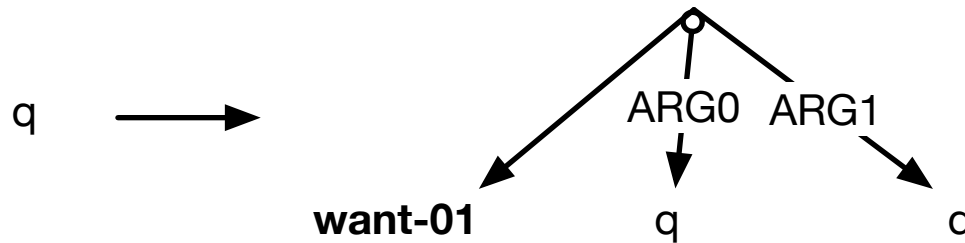
## Grammar rules (some of them)



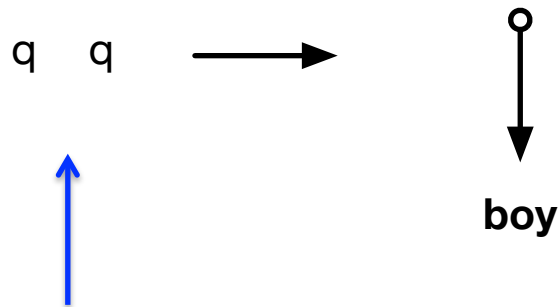
(figure from Chiang et al, 2013)

# DAG Automata

(Kamimura and Slutzki, 1981. Quernheim and Knight, 2012)



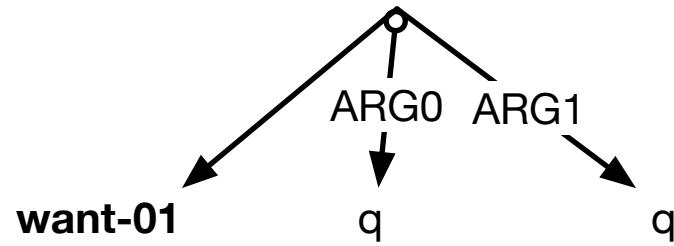
Rewrite rule  
("explicit rule")



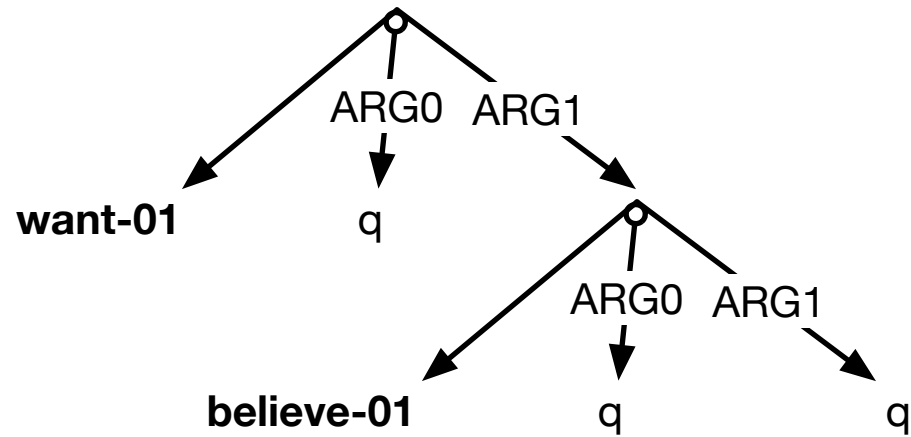
Merge rule  
("implicit rule")

Two or more states can merge rule

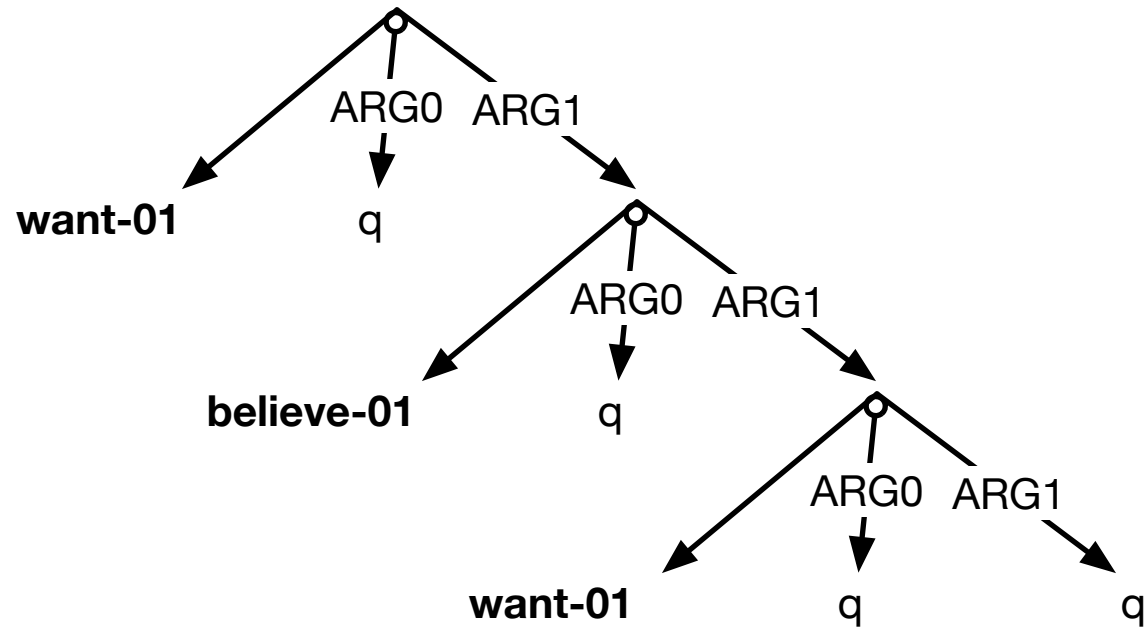
# DAG Automata



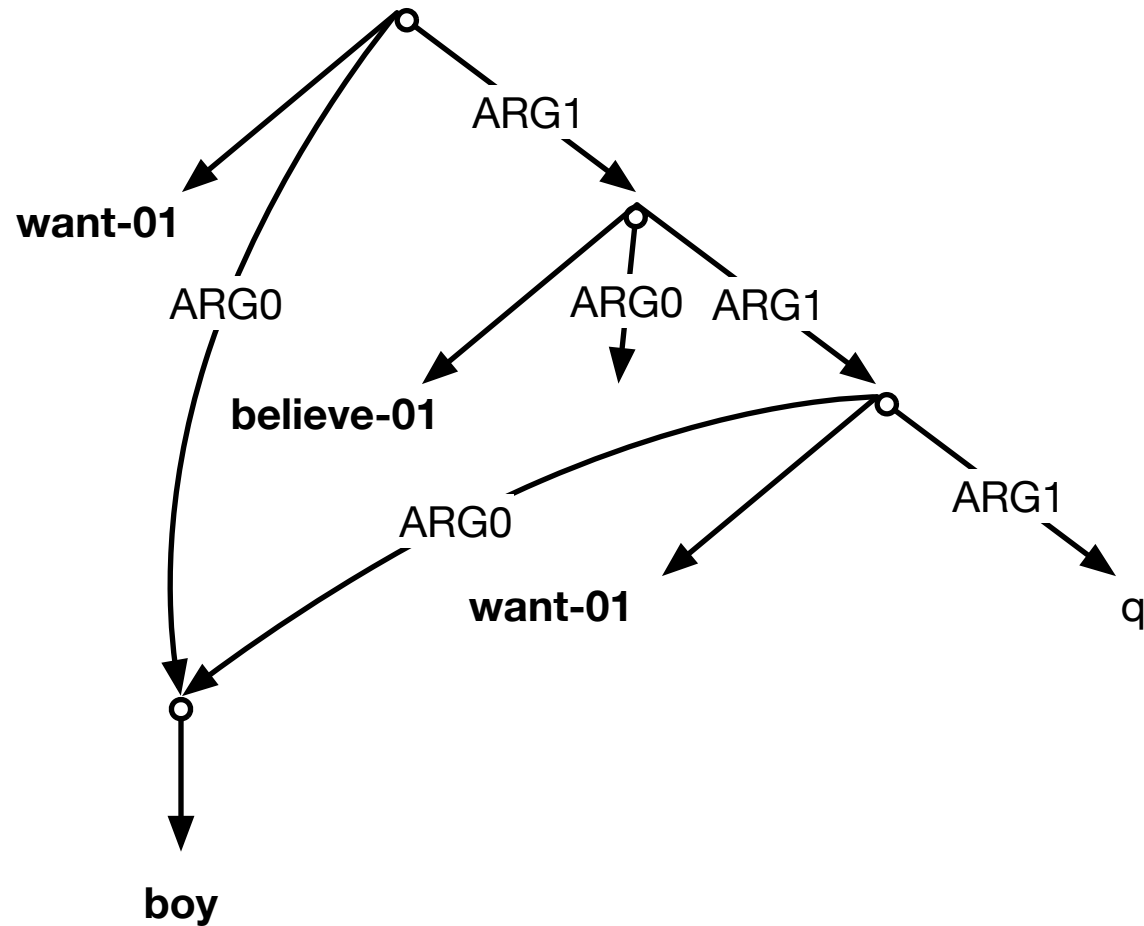
# DAG Automata



# DAG Automata

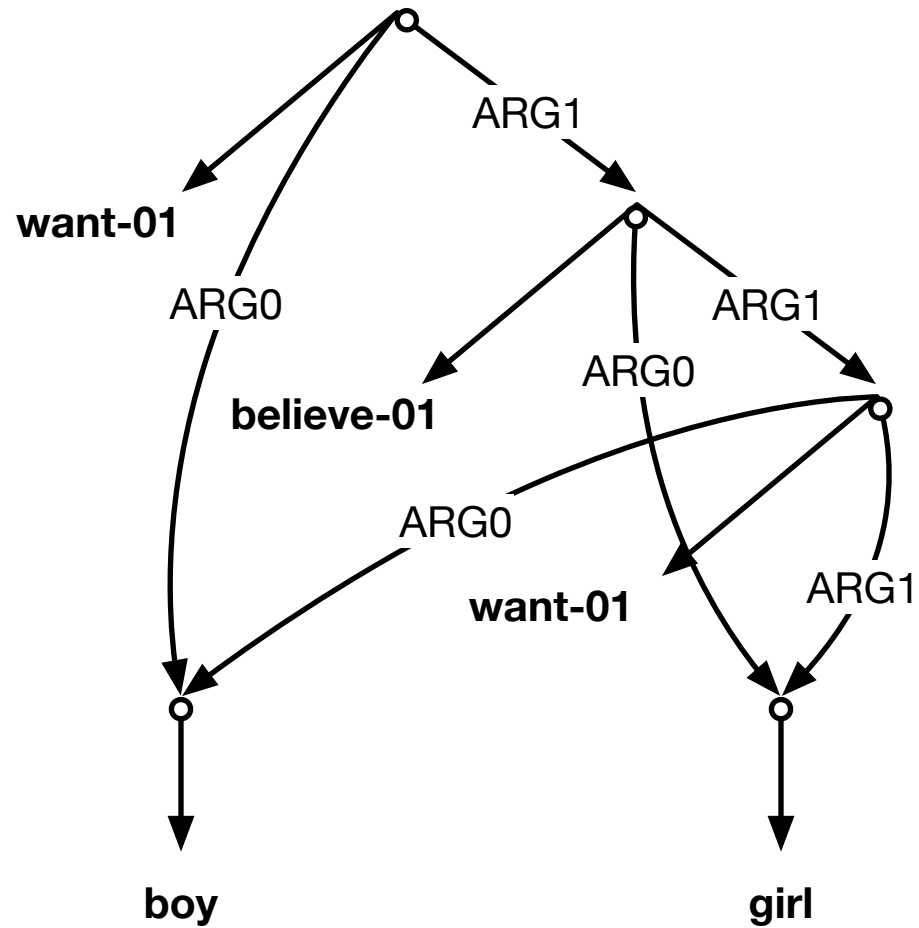


# DAG Automata





# DAG Automata



# Extensions

- Weighted and probabilistic grammars
- Synchronous grammars and transducers
  - Useful for building parsers, generators, and MT systems

## Recent/Ongoing work

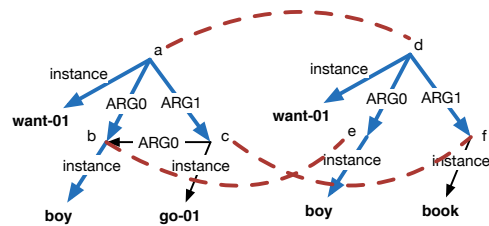
- Improved parsing algorithms (Chiang et al, 2013)
- Applications to parsing and generation (Braune et al, 2014) and MT (Jones et al, 2012)
- Implementations
  - Hyperedge replacement grammars: **Bolinas** (Chiang et al, 2013; Jones et al, 2012)
  - DAG automata: **DAGGER** (Quernheim & Knight, 2012)

# Alignment

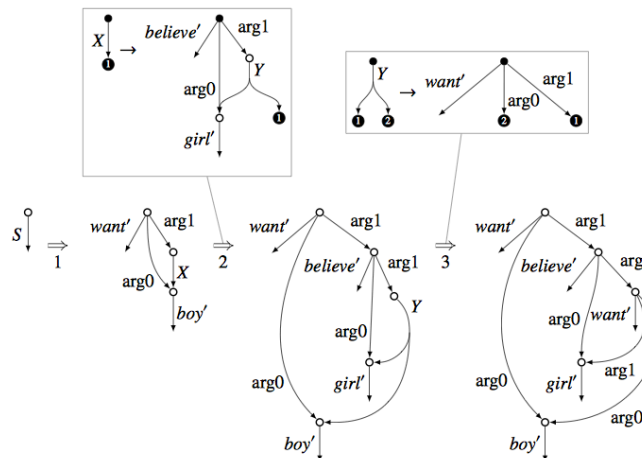
IAEA accepted North Korea's proposal in November.

```
(a / accept-01
  :ARG0 (o / organization
    :ARG0 (n / name
      :op1 "IAEA"))
  :ARG1 (t2 / thing
    :ARG1-of (p / propose-01
      :ARG0 (c / country
        :name (n2 / name
          :op1 "North"
          :op2 "Korea"))))
  :time (d / date-entity
    :month 11))
```

# Evaluation

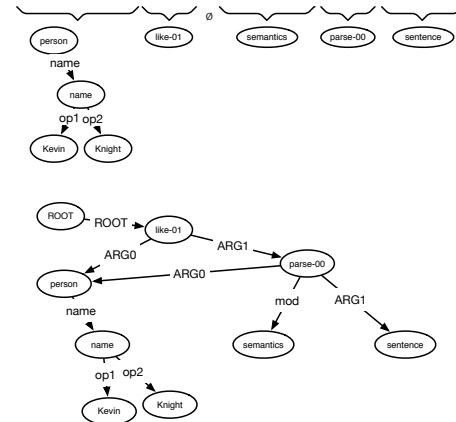


# Graph grammars

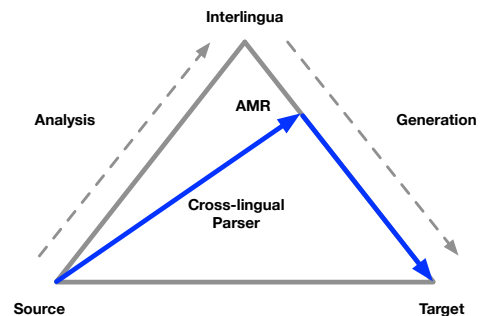


# Parsing

Kevin Knight likes to semantically parse sentences



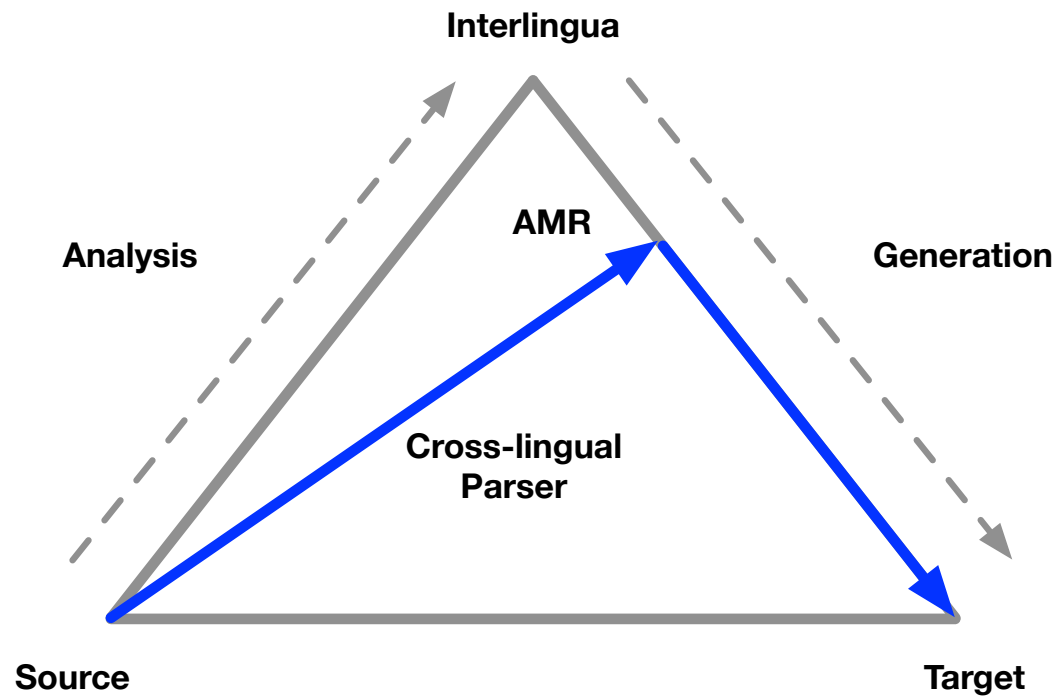
# Applications



# Applications

- Alignment
- Parsing
- Evaluation
- Graph Grammars and Automata
- **Applications**
  - MT, Summarization, Entity linking

# Machine Translation

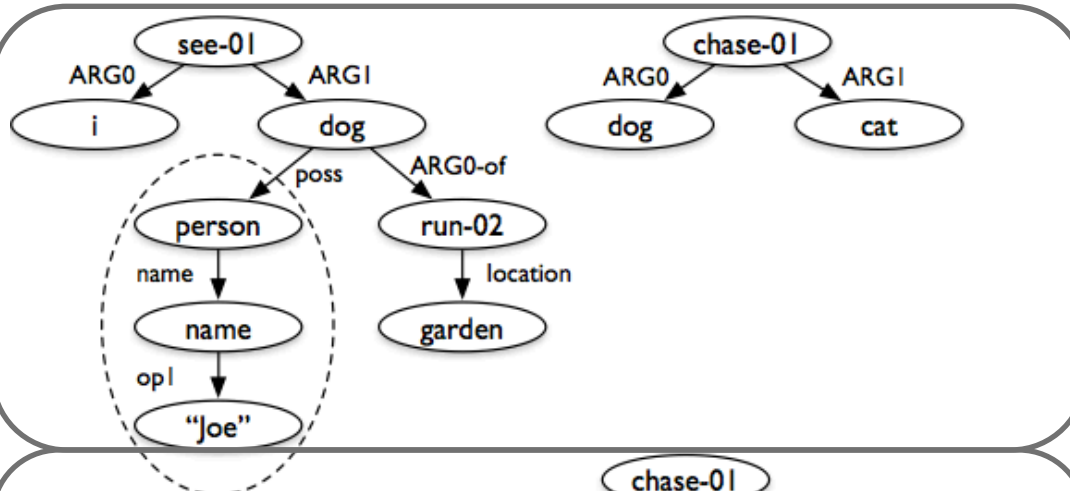


# Summarization (Liu et al, NAACL 2015)

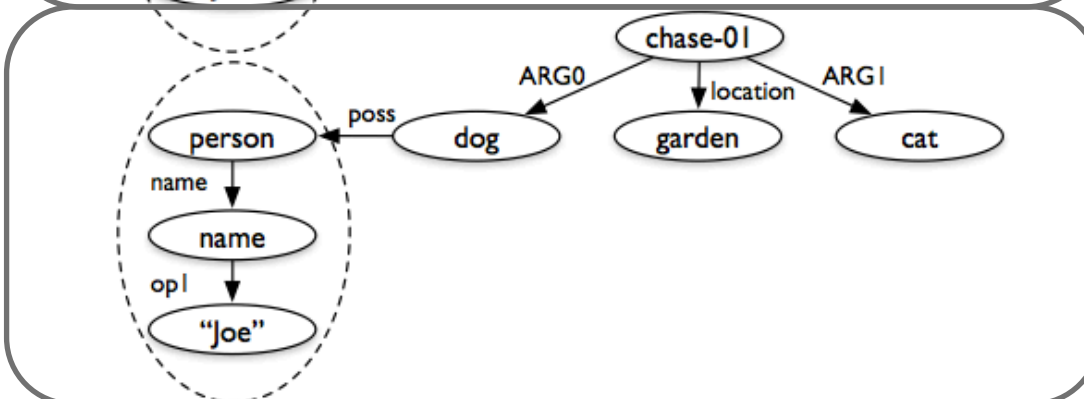
Document Sentences  
(input)

Sentence A: I saw Joe's dog, which was running in the garden.  
Sentence B: The dog was chasing a cat.

Document AMRs  
(run parser)



Summary AMR  
(select nodes  
and edges)

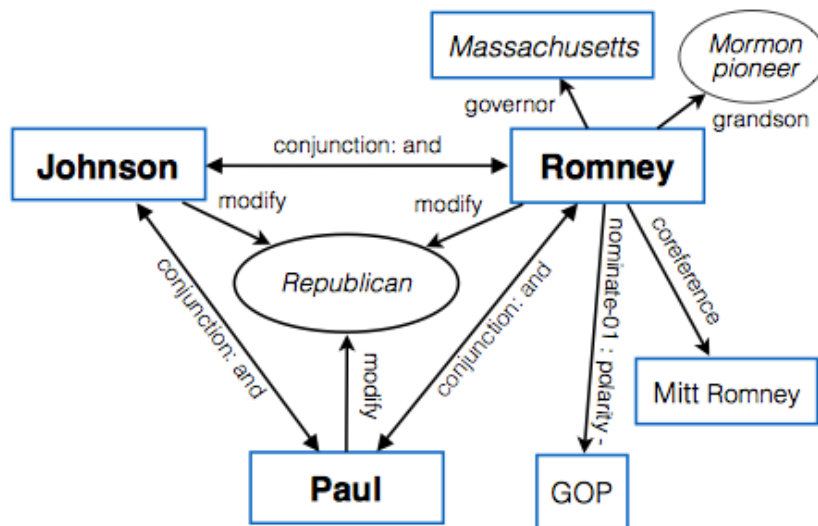


Summary (generate)

Summary: Joe's dog was chasing a cat in the garden.

# Unsupervised Entity Linking with AMR (Pan et al, NAACL 2015)

- Link entity mentions in text to knowledge base
- Look at context to disambiguate mention
- Uses AMR graphs as context to build knowledge networks:



AMR context performs much better than SRL context for unsupervised entity linking



# AMR at NAACL 2015

- Talks
  - Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, Noah A. Smith. *"Toward Abstractive Summarization Using Semantic Representations"*
  - Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, Kevin Knight. *"Unsupervised Entity Linking with Abstract Meaning Representation"*
- Posters
  - Chuan Wang, Nianwen Xue, Sameer Pradhan. *"A Transition-based Algorithm for AMR Parsing"*
- Demonstrations
  - Lucy Vanderwende, Arul Menezes and Chris Quirk. *"An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus"*
  - Naomi Saphra and Adam Lopez. *"AMERICA: an AMR Inspector for Cross-language Alignments"*

# Resources

- **AMR website:** <http://amr.isi.edu>
- **JAMR:** <https://github.com/jflanigan/jamr>
- **Transition-based parser:**  
<https://github.com/Juicechuan/AMRParsing/>
- **Bolinas toolkit:**  
<http://www.isi.edu/publications/licensed-sw/bolinas/>
- **DAGGER toolkit:**  
<http://www.ims.uni-stuttgart.de/~daniel/dagger>

(t / thank-01  
:ARG1 (y / you) )

Thanks to: Miguel Ballesteros, David Chaing,  
Shay Cohen, Chris Dyer, Kevin Knight, Lingpeng  
Kong, Fei Liu, Noah Smith, Sam Thomson, and  
Chuan Wang